Report No. 370-3

[redacted]

SOUPAC PROGRAM DESCRIPTIONS

STATISTICALLY ORIENTED USERS PROGRAMMING AND CONSULTING

March 1, 1971

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

/Soupac Program Description/

This manual has been prepared by the Statistical Consultants of the Department of Computer Science, University of Illinois (Urbana Campus), as documentation for the system of statistical programs known as SOUPAC. SOUPAC has been written by the statistical consultants of this department in an effort to provide a broad range of standard statistical procedures which are of use to the academic community at large. Inquiries about SOUPAC should be directed to 138 Digital Computer Laboratory.

PROGRAM LIST

February, 1971

AUTOCORRELATION AND SPECTRAL ANALYSIS
BALANOVA 5
BINORMAMIN
BISERIAL CORRELATION
CANONICAL ANALYSIS
CENTROID FACTOR ANALYSIS
CLASSIFICATION
CLIQUE ANALYSIS
COMMUNALITY ESTIMATION
CORRELATION
DISCRIMINANT ANALYSIS
ECONOMETRIC REDUCED FORM AND RESIDUAL ANALYSIS
FREQUENCY COUNTING - MEASURES OF ASSOCIATION
FIT (CHI SQUARE GOODNESS-OF-FIT TEST)
ITERATIVE FACTOR ANALYSIS
JACOBI (Eigenvalues and Eigenvectors)
K - CLASS ESTIMATION
KOLMOGOROV-SMIRNOV
LINEAR PROGRAMMING
MATRIX OPERATIONS
MISSING DATA CORRELATION
MULTIPLE CORRELATION
OBLIMAX
PAIRED COMPARISONS
PARTIAL CORRELATION
PRINCIPAL AXIS FACTOR ANALYSIS
PROBIT (Maximum Likelihood Regression)
PROCRUSTES
QUADRATIC PROGRAMMING
RANDOM NUMBER GENERATOR
RANKING
SCALOGRAM ANALYSIS
SQUARE ROOT FACTOR ANALYSIS
STANDARD SCORES
STEP-WISE MULTIPLE CORRELATION
THREE MODE FACTOR ANALYSIS
THREE-STAGE LEAST SQUARES
TRANSFORMATIONS
T-TESTS
UNRESTRICTED MAXIMUM LIKELIHOOD FACTOR ANALYSIS
VARIMAX

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

INTRODUCTION TO SOUPAC
A USER'S GUIDE

A SOUPAC program consists, at least, of (1) IBM 360 system statements, and (2) SOUPAC statements. The IBM 360 system statements are used to give instructions to the 360 computer, and the SOUPAC statements are used to describe the types of data manipulations and statistical analyses that are to be performed. The set of SOUPAC statements in one job is called a SOUPAC parameter deck, and it will include references to one or more individual SOUPAC programs. Note all cards should be punched on an IBM 029 keypunch.

Every 360 system statement must be on a separate card and must have the characters // or /* in columns 1 and 2. In particular, the system cards for a SOUPAC job are listed here, in the order they must appear:

```
CARD 1      /*ID      accounting information

CARD 2      //  EXEC  SOUPAC

CARD 3      //SYSIN  DD  *
                      SOUPAC parameter deck

CARD 4      /*
```

Note that these four cards are an absolute minimum.

CARD 1

The /*ID card is punched on a yellow striped card available in the keypunch area. This card has no clipped corners and is used only as an ID card for jobs to be run on the 360. The first five columns of this card must contain the characters /*ID followed by a blank. This is the only card in your deck which must be punched on a special card; the rest of your deck may be punched on any of the standard corner cut cards found in the keypunch area.

Sample ID cards:

```
/*ID PS=9999,DEPT=AGRON,NAME=SMITH
/*ID NAME='BOB_SMITH',PS=8899,DEPT=VOTEC,LINES=5000
/*ID CODE=SWITCH,PS=9988,NAME=STEPPENWULF,DEPT=REC
```

The accounting information consists of the following keywords and responses:

	KEYWORD	RESPONSE
necessary	PS=	your problem specification number
	DEPT=	your department
	NAME=	your name

	KEYWORD	RESPONSE
optional	(min,sec) TIME=(,sec) min	the optional TIME parameter on the ID card is the estimate of execution time for the job. By default it is 1 minute. The time estimate on the ID card is the time estimate for the SOUPAC job. If there is a time estimate on the // EXEC SOUPAC card there must be at least as much time on the ID card.
	IOREQ=n	where n is the number of INPUT/OUTPUT requests. The default is 1000. Moving from cards to a temporary storage location generates one I/O request per observation. Moving to or from a temporary storage location generates two I/O requests per observation. Cards read in or lines printed out do not generate I/O requests. The interactive nature of SOUPAC programs often necessitates considerable amounts of data transfer. For this reason, I/O requests for SOUPAC jobs are relatively large.
	CODE=ZZZZZ	code is the signal that your problem specification number is code word protected. If you have code word protection, get the code word from your approved problem specification request form. If you do not have code word protection, do not use the keyword CODE.
	LINES=XXXXX	you requested XXXXX lines of output. The default is 2000.
	CARDS=XXXX	you requested XXXX cards punched. The default is NONE.

There must be no blank spaces in the accounting information and the order of the accounting information makes no difference. Blanks in a name, such as JOHN USER, are not allowed. The name must be only one word or of the form JOHNUSER. If blanks are desired, then the name must be enclosed in apostrophes and an underscore substituted for the blank, e.g. 'JOHN_USER'.

Note further that commas separate keywords from responses to keywords and that the equal sign is a part of the keyword response. Furthermore, your ID card parameters may not go beyond column 71. To continue an ID card put a comma after the last parameter that will fit completely into 71 columns or less and proceed on a second card which has the characters /* punched in columns 1 and 2, followed by at least one blank. This second card may be any standard corner cut card.

Further information about additional parameters is available from either the Service Programming Area or the SOUPAC office.

CARD 2

The // EXEC SOUPAC is punched as in the example on page 1 with two slashes in columns 1 and 2, followed by at least one blank. This is followed by the word EXEC, at least one blank, and then the word SOUPAC.

There is in fact another way to get a SOUPAC job. One can say:

```
// EXEC SOUP
//SYSIN DD *
```

The essential differences are these: SOUPAC has 15 available temporary storage locations; SOUP has only 5 temporary storage locations. SOUP, however, runs about five seconds faster than SOUPAC and is generally recommended for users who do not require more temporary storage locations than S1 through S5.

The amount of memory to be used is specified on each // EXEC SOUPAC or // EXEC SOUP card rather than on the ID card. SOUPAC jobs are given a default region of 150K which is adequate for nearly all SOUPAC jobs. See sections on options to a SOUPAC job.

CARD 3

The next card is copied as in the example with //SYSIN beginning in column 1 and extending through 7 without any blanks. This is followed by at least one blank, the two characters DD, at least one blank, finally the character *.

SOUPAC Statements

The SOUPAC parameter deck is divided into two main sections. The first consists of the problem program parameter cards. Individual problem program write-ups are available to users through the SOUPAC office. The last card of the problem program parameter cards must always contain the words END SOUPAC as the first non-blank characters. The END SOUPAC card is immediately followed by any data decks you may have. All data decks are preceded by a DATA format card and followed by an END# card. The format card has the word DATA as the first four non-blank characters. After the word DATA may be any comments the user may want, if any. These comments will be printed on the user's output, but will be otherwise ignored by SOUPAC. Next comes the data card parameters. The user must specify the number of variables per row of data to be input, and also may optionally specify the number of rows of data(observations). Following this information is a standard FORTRAN type format enclosed in parentheses. This is the format which describes the structure of the data deck it precedes. The following forms of the data format card are valid:

```
DATA(observations,variables)(format string≤592 non-blank characters)

DATA(,variables)(format string≤592 non-blank characters)

DATA(variables)(format string≤592 non-blank characters)
```

Remember that all data decks must be terminated by END# as the first four non-blank characters on a single card.

Data used for calculations should be read in either E or F format. Data not used for calculations can be read in any format.

END CARDS

Within a SOUPAC parameter deck, three types of END cards are used:

END#: This card is used to end each data deck which appears. For each DATA format card used, there must be a corresponding END# card at the end of the data deck.

END S: This card appears once and only once per SOUPAC parameter deck and must appear as the last card in the parameter deck. Its function is to separate the SOUPAC parameter deck from the data decks.

END P: This card is used to indicate the end of any program which requires subparameter cards and must appear for all such programs. (In this context $-control cards are not taken to be subparameter cards). Any program which needs the END P card explicitly states so within its individual program write-up. For all other programs, use of the END P card is an error. If a program which uses the END P card is also the last program to be executed it must be terminated by an END P card then an END S card must appear to indicate that no more cards follow.

CARD 4

The last physical card in your deck must contain the characters /* in columns 1 and 2. No other punches should appear on this card.

Sample SOUPAC deck setup:

```
//ID NAME=SMITH,DEPT=PEM,PS=6894,LINES=5000,CARDS=500,IOREQ=10000
//  EXEC  SOUPAC
//SYSIN  DD  *
    program 1 ( )( )( )( ).....                 .
    program 2 ( )( )( )( ).....                 .
        :
    program n ( )( )( )( ).....                 .
END SOUPAC          .
DATA
 :
END#
/*
```

Note that there may be more programs than data decks or more data decks than programs. In any case, for programs that read data decks, the user must insure that the order of data decks corresponds to the order of programs that read data decks since the first program to read a data deck will read the first available deck, and so on. The process continues until either the SOUPAC program list is exhausted or a program reading data cannot find a data deck available. Notice that extra data decks are ignored.

A FINAL NOTE

Any changes the user may want to specify in the job control language of SOUPAC must follow the standard OS/360 conventions.

OPTIONS TO A SOUPAC JOB:
PARMS, PROLOG CARDS, AND $-CONTROL CARDS

A. PARMS

Parms are arguments to the keyword 'Parameter,' contracted into the keyword 'PARM', which give instructions to a processor running under a 360-system. In this context, SOUPAC is a processor running under a 360 system. PARMS are always coded on an EXEC card and have the following form:

```
// EXEC  SOUPAC,PARM='OPT1,OPT2,....,OPTm'
```

The permissable options to be used as SOUPAC PARMS are listed below with an explanation of their use and function. Note that the default is underlined, that is, // EXEC SOUPAC is equivalent to // EXEC SOUPAC,PARM='OPT1,OPT2...' where the underlined PARM is to be taken as one of the list of options in the PARM string in the example. These PARMS give the SOUPAC system instructions in the same way that parameters give SOUPAC statistical or data management programs instructions.

1. NODYNAM or DYNAM

   NODYNAM implies that a non-dynamically allocatable version of the library of statistical procedures is to be used. This version will run in some 150K of core and will handle a lesser number of variables than the dynamically allocatable version. DYNAM will use the dynamically allocatable version of any program requested which will handle more variables in an arbitrarily specified amount of core above a certain minimum. If using DYNAM, see the SOUPAC consultants for a handout on optimal region sizes for particular numbers of variables.

2. EXECUTE or NOEXECUTE

   NOEXECUTE implies that the SOUPAC parameter deck, for which the Syntax Interpreter is to scan and build intermediate parameters, should not be executed. NOEXECUTE indicates that only a syntax check is to be performed. If EXECUTE is specified and no errors are found by the Syntax Interpreter, the job step will proceed. If EXECUTE is specified and errors are found by the Syntax Interpreter, execution of the step may continue depending upon whether LET or NOLET is also specified.

3. NOLET or LET

   If an error is found by the Syntax Interpreter and EXECUTE has been specified, execution will proceed only if LET was also specified. In this case, execution will proceed only through the last program processed which was completely error free. If NOLET was specified and errors are found by the Syntax Interpreter, execution will not be permitted.

4. LIST or NOLIST

   LIST indicates that all program cards are to be listed. NOLIST indicates that only the prolog section of the SOUPAC parameter deck is to be listed.

5. PGM or NOPGM

PGM indicates that a complete SOUPAC parameter deck and data decks follow. NOPGM indicates that only the prolog section and data deck follow, and that the intermediate parameters are being provided by the user by over-riding the cataloged procedure. This implies that the user has previously run a SOUPAC job and has saved the two necessary data sets so that he may run the same program again. To perform this saving of data sets correctly, a user should visit the SOUPAC office first to ensure it is done correctly.

If any error is found by the Syntax Interpreter in the prolog section, the job step will not continue.

If the job step which generated the intermediate parameter data sets found syntax errors, execution of the job step in which NOPGM is specified will continue (If EXECUTE is specified) through the last program processed which was completely error free regardless of whether LET or NOLET was specified in either job step.

Examples:

To do just a syntax check:

```
//  EXEC  SOUPAC,PARM='NOEXECUTE'
```

To execute up to the first program found to have syntax errors:

```
//  EXEC  SOUPAC,PARM='LET'
```

To execute up to the first program found to have syntax errors and use the dynamically allocatable library:

```
//  EXEC  SOUPAC,PARM='LET,DYNAM'.
```

Note that the PARMS may be listed in any order.

B. PROLOG OF A SOUPAC JOB

Described below are several # control cards which may appear in the prolog of a SOUPAC job. Within the prolog these control cards may appear in any order. If prolog control cards are used, they must appear immediately after the SYSIN card. The Syntax Interpreter determines the end of the prolog when it reads a card which is not one of these types. All types have parameters and must be terminated by a period. Prolog cards may not have continuation cards, hence all parameter information must be punched within 80 columns. There is no limit to the number of prolog cards permitted nor is there any restriction on the number of any one type. If conflicting information is entered, the information entered last overrides any previous definitions.

1. #REPEAT OPTION

The #REPEAT OPTION is used to repeat sections of a SOUPAC parameter deck an optional number of times. The #REPEAT card which appears in the prolog section will be followed by up to 22 (twenty-two) integer parameters which will indicate the number of repetitions of up to 22 repeat sequences. The card sequences to be repeated will be preceded and followed by #SREP and #EREP cards respectively. Example:

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
#REPEAT (2).
<additional program cards>
#SREP
CORRELATION (C)( )(S1).
SQUARE ROOT FACTOR ANALYSIS (S1)(P(F))(20)(C)(P(F)).
#EREP
END S
```

In this example the program sequence of CORRELATION and SQUARE ROOT FACTOR ANALYSIS will be repeated twice. Four card input data sets would be required for the repeated sections.

Repeat sequences which begin before a main program and end in a subprogram or which begin in a subprogram and do not end in the same subprogram are not allowed. Nested or overlapping repeat sequences are not allowed. Also a #SREP card cannot be immediately followed by a #EREP card and a single appearance in the deck of either card will cause an error.

2. #V-UNIT OPTION

#V-UNIT allows the user to change input and output addresses in the execution of one SOUPAC job. The form of a #V is as follows:

$$\#Vn\ (m)\ (A_1).\ .\ .\ .\ .\ .\ .\ .\ .(A_k).$$

where n is an integer 1 through 9, thus there can be at most 9 variable addresses, namely V1 through V9; and m is a counter which determines how many times a variable address may be used before it assumes the next value in its list of possible values. $A_1 .....A_k$ are addresses which Vn assumes. These can be any valid address. At the moment, however, forms like (S1/P) will not work. Note that , CARDS and PRINT are permitted.

Finally, the list of addresses is cyclic; that is, if, after A_k has been used, Vn occurs again in the program, Vn will have the value A_1, and so on.

```
/*ID
//  EXEC SOUP
//SYSIN  DD  *
#V9(1)(S1)(S2)(S3)(S4).
#V5(1)(S1)(S2)(S3)(S4).
#REPEAT (4).
MAT.
```

(Example, Continued)

```
#SREP
MOV (C)(V9).
#EREP
HOR (V5)(V5)(V5)(V5)(S5).
END P
  .
  .
  .
END S
```

This program segment reads 4 separate card decks, saving them in temporary storage, and horizontally augments them into one data set.

The equivalent without the use of #REPEAT and #V would be as follows:

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
MAT.
MOV(C)(S1).
MOV(C)(S2).
MOV(C)(S3).
MOV(C)(S4).
HOR(S1)(S2)(S3)(S4)(S5).
END P
   .
   .
   .
   .
END S
```

Note that the MOV(C)(V9). statement is expanded into four move statements and V9 takes the values S1 through S4. Similarly, V5 takes on the values of S1 through S4.

3. #OLD OPTION

The #OLD option is used to define the number of rows in a sequential data set created by a previously run SOUPAC job. The number of rows is then entered into a table in the monitor. This option should be used whenever the header record on the data set is not known to have a correct value for the number of rows, and the user does not want to execute a MATRIX MOVE to count the rows. To use the option, punch a card with #OLD in the first four columns. Then code the address and the number of rows in the usual SOUPAC fashion. The number of columns may be coded on the card if desired, but will be totally ignored. Include this card in the prolog section of the SOUPAC job.

For example, to indicate that a data set to be input from SEQUENTIAL 1 has 77 rows you would prepare the following card:

```
#OLD (S1)(77).
```

4. #TEST OPTION

There is also available a #TEST option; however, this facility is complicated and intended for testing purposes within the SOUPAC office and has no significant advantage for the general user.

5. #DEFINE OPTION

Whenever the user wishes to specify the dimensions of a direct access data set (DISK address), punch #DEFINE in the first seven columns of a card followed by the address, number of rows and number of columns coded in the usual SOUPAC fashion. Include this card in the prolog section of your program. For double precision matrices, code the same number of rows, but twice as many columns as otherwise. DISK 1 and DISK 2 have default definitions of 450 rows by 450 columns single precision. If the user desires any other dimensions on these data sets, #DEFINE must be used. If the user desires to use any DISK address other than DISK 1 and DISK 2, #DEFINE must be used besides supplying the necessary DD cards.

For example, to define a data set for DISK 17 with 20 rows and 40 columns double precision, you would prepare the following card:

```
#DEFINE (DISK 17)(20)(80).
```

Notice that all prolog cards start with a # in column one and must occur before any SOUPAC program parameter cards. A #-card in the middle of the SOUPAC program parameter deck is treated as a comment. There is, however, a #-control card, while not strictly a prolog card, which may occur in the SOUPAC program parameter deck and will not be treated as a comment. This is the #-zero card and is the only exception to the statement about # cards being comments if in the middle of the deck. The #-zero card is essentially a debugging tool to facilitate reading of dumps if one is needed. It has no particular use for the user.

C. $-CONTROL CARDS

$-CONTROL CARDS are used to provide additional information to a SOUPAC program above and beyond what is included in the parameters. There are 3 $-control cards. All must begin in column one with the character $ and then continue accross the card without blank columns.

1. $C-B

The $C-B card provides as its arguments the variables to be used as control breaks for a program which accept control breaks. The use of this card with a program which does not accept control breaks is an error. The form of this card is as follows:

$C-B(V_1)(V_2).....(V_n).

When V_1 through V_n are variable numbers and n must be less than or equal to 24.

2. $INP

$INP has as its arguments a string of input addresses. The form is:
$INP($A_1$).......($A_n$).
where A_1 through A_n are input addresses including cards. The number of addresses will be determined by the program accepting the $INP card and will explicitly mentioned in the program write-up.

3. $OUT
$OUT($A_1$).......($A_n$).
$OUT has as its arguments a string of output addresses. The form is the same as that for $INP and the number of addresses is also determined by the program accepting the $OUT card. Multiple output address will be accepted. See section on Input/Output multiple addresses.

SOUPAC INPUT-OUTPUT AND TEMPORARY STORAGE

I. GENERAL

A. Input and Output as Data Types

Consider a set of data which a researcher wants intercorrelated. To do correlations there is in the SOUPAC library of statistical precedures a correlation program. Input to the correlation program is the researcher's raw data; output from the correlation program is a matrix of correlation coefficients. Similarly, every conceivable program has a particular input; in fact, perhaps several inputs, and some output.

The nature of the input and output of a particular program will depend on the program and its intent. For example, raw data variables are input into a correlation program which outputs a correlation matrix. But a factor analysis program expects as input a correlation matrix, and yields as output a factor matrix. In contrast to the singular relation of the nature of input and output to a particular statistical program, every program finds its input somewhere and must put its output somewhere.

B. Input and Output as Data Sources

SOUPAC is designed in such a manner that the researcher can tell any program where his inputs are and where to put his outputs. Punched cards are an obvious input source; printed pages are an obvious output source. But the nature of a punched card deck input into a correlation program would be that of raw data variables. In the SOUPAC system input and output sources are also called addresses. Thus, a possible input address for a correlation program is cards and a possible output address for correlation coefficients is print. Input and output addresses are parameters to every program in the SOUPAC system. As the researcher reads a particular program write-up he will notice that the order of the parameters determines the nature of his input or output and his supplying an input or output address determines whether or not he uses or gets the particular inputs and outputs.

II. ELEMENTARY INPUT/OUTPUT ADDRESS AND TEMPORARY STORAGE

A. Possible elementary input and output addresses in the SOUPAC system are these:

INPUT: CARDS, SEQUENTIAL 1, SEQUENTIAL 2, . . .
. . . SEQUENTIAL 15

OUTPUT: PRINT, SEQUENTIAL 1, SEQUENTIAL 2, . . .
. .SEQUENTIAL 15 (See section on punched cards).

Again, CARDS and PRINT are obvious sources. SEQUENTIAL 1 through SEQUENTIAL 15, however, are input or output names of 15 temporary storage regions available to the researcher in the SOUPAC system. These 15 temporary storage regions are provided for exactly that purpose, temporary storage of data. Notice that with this facility a user can save his correlation matrix, for example, at SEQUENTIAL 1 and then give SEQUENTIAL 1 as an input address to a factor analysis program. Or a researcher can construct a copy of his data on temporary storage and then let any number of programs use the same data as input from the same input address, saving him the effort of making multiple copies of his card deck so that each program would read its own deck. Finally, temporary storage addresses enable the saving of intermediate results for further processing or modification by other programs and thereby enable the researcher to construct his own analysis procedure by providing the appropriate inputs and outputs to the right programs at the right times.

B. SOUP vs SOUPAC with Respect to Temporary Storage

There are two ways of invoking the SOUPAC system. One can ask for SOUPAC or SOUP. Note that all 15 temporary storage regions are allocated to SOUPAC, while only SEQUENTIAL 1 through SEQUENTIAL 5 are allocated to SOUP. Asking for SEQUENTIAL 6 through SEQUENTIAL 15 when running under SOUP will cause an error and terminate the job.

All of these input-output addresses may be abbreviated as follows:

CARDS	C
PRINT	P
SEQUENTIAL 1	S1 (or T1)
.	
.	
.	
SEQUENTIAL 15	S15 (or T15)

T1 through T15 are alternative abbreviations for SEQUENTIAL 1 through SEQUENTIAL 15. T1 through T15 are, in fact, abbreviations of TAPE 1 through TAPE 15. SEQUENTIAL 1 through SEQUENTIAL 15 and their abbreviations are the recommended uses. The T1 through T15 notation reflects a real technical distinction but has been kept to enable programs using that notation to run.

C. Multiple Output Addresses

A researcher may want to output to several sources: he may desire to both print and save some results for later use. He cannot, however, input from more than one source for a particular input address. The facility of multiple output addresses has the following construction:

(output address^1/output address^2/output address^3).

This is the completely general form providing for up to three separate outputs. Each output must be a different source, however. Thus, (S1/P/X) is a valid multiple output address providing for temporary storage at S1, a punched copy of the data. (See section on punch for explanation of X). (S1/P) will print and store but not punch. The order of the addresses makes no difference. (S1/P) is equivalent to (P/S1). Forms such as (S1/S2), however, are not permitted, nor are (P/P) or (X/X): one can output only to one sequential and only once to P or X.

The above general form is available only if the output address in the particular program is marked with an Ω.

In all cases, however, the form

(output address 1/output address 2)

is valid unless the program write-up explicitly has a restriction.

D. Print is F Form of Output Print Address.

There is yet another form to output addresses. This form is available only where the researcher finds the symbol Ω in the program write-up and has to do with the kind of printed output. For technical reasons, most programs print in a form called E-format which is a form of scientific notation. This form allows the computer to print numbers of any size. Some programs, for which the output numbers are known to be constrained, as in correlation coefficients, however, print in a form called F-format which is ordinary decimal number representation. F-format generally cannot print numbers larger than a pre-determined size. The size of number depends on the nature of a researcher's data, but the program has no way of knowing this, hence, the most general form, E-format is used.

The researcher however, can on option specify F-format. To print in F-format he would use the following output address:

(P(F)) or (P(F)/S1) if he wanted a multiple

output address. Those programs which print in F-format already, as for correlation coefficients, can be made to print in E-format by using the following output address:

(P(E)) or (P(E)/S15).

The different forms look like this:

E-format	Scientific Notation	F-format	Decimal Number Representation
± 0.12345E 06	± 1.2345 x 10^5	± 123456.12345	±123456.12345

All four numbers have the same value correct to 5 places. Notice that F-format cannot represent a number greater than 999999.99999 in absolute value whereas E-format can represent the first 5 digits of any number of order of magnitude up to 10^{99} . The numbers of digits illustrated for E and F formats are the pre-determined limits for the size of numbers. E-format is the more general form but F-format is easier to read.

correlation matrix, for example, at SEQUENTIAL 1 and then give SEQUENTIAL 1 as an input address to a factor analysis program. Or a researcher can construct a copy of his data on temporary storage and then let any number of programs use the same data as input from the same input address, saving him the effort of making multiple copies of his card deck so that each program would read its own deck. Finally, temporary storage addresses enable the saving of intermediate results for further processing or modification by other programs and thereby enable the researcher to construct his own analysis procedure by providing the appropriate inputs and outputs to the right programs at the right times.

B. SOUP vs SOUPAC with Respect to Temporary Storage

There are two ways of invoking the SOUPAC system. One can ask for SOUPAC or SOUP. Note that all 15 temporary storage regions are allocated to SOUPAC, while only SEQUENTIAL 1 through SEQUENTIAL 5 are allocated to SOUP. Asking for SEQUENTIAL 6 through SEQUENTIAL 15 when running under SOUP will cause an error and terminate the job.

All of these input-output addresses may be abbreviated as follows:

CARDS	C
PRINT	P
SEQUENTIAL 1	S1 (or T1)
.	
.	
.	
SEQUENTIAL 15	S15 (or T15)

T1 through T15 are alternative abbreviations for SEQUENTIAL 1 through SEQUENTIAL 15. T1 through T15 are, in fact, abbreviations of TAPE 1 through TAPE 15. SEQUENTIAL 1 through SEQUENTIAL 15 and their abbreviations are the recommended uses. The T1 through T15 notation reflects a real technical distinction but has been kept to enable programs using that notation to run.

C. Multiple Output Addresses

A researcher may want to output to several sources: he may desire to both print and save some results for later use. He cannot, however, input from more than one source for a particular input address. The facility of multiple output addresses has the following construction:

(output address^1/output address^2/output address^3).

This is the completely general form providing for up to three separate outputs. Each output must be a different source, however. Thus, (S1/P/X) is a valid multiple output address providing for temporary storage at S1, a print of the same data, and a punched copy of the data. (See section on punch for explanation of X). (S1/P) will print and store but not punch. The order of the addresses makes no difference. (S1/P) is equivalent to (P/S1). Forms such as (S1/S2), however, are not permitted, nor are (P/P) or (X/X): one can output only to one sequential and only once to P or X.

The above general form is available only if the output address in the particular program is marked with an Ω.

In all cases, however, the form

(output address 1/output address 2)

is valid unless the program write-up explicitly has a restriction.

D. Print is F Form of Output Print Address.

There is yet another form to output addresses. This form is available only where the researcher finds the symbol Ω in the program write-up and has to do with the kind of printed output. For technical reasons, most programs print in a form called E-format which is a form of scientific notation. This form allows the computer to print numbers of any size. Some programs, for which the output numbers are known to be constrained, as in correlation coefficients, however, print in a form called F-format which is ordinary decimal number representation. F-format generally cannot print numbers larger than a pre-determined size. The size of number depends on the nature of a researcher's data, but the program has no way of knowing this, hence, the most general form, E-format is used.

The researcher however, can on option specify F-format. To print in F-format he would use the following output address:

(P(F)) or (P(F)/S1) if he wanted a multiple

output address. Those programs which print in F-format already, as for correlation coefficients, can be made to print in E-format by using the following output address:

(P(E)) or (P(E)/S15).

The different forms look like this:

E-format	Scientific Notation	F-format	Decimal Number Representation
± 0.12345E 06	± 1.2345 x 10^5	± 123456.12345	±123456.12345

All four numbers have the same value correct to 5 places. Notice that F-format cannot represent a number greater than 999999.99999 in absolute value whereas E-format can represent the first 5 digits of any number of order of magnitude up to 1099 . The numbers of digits illustrated for E and F formats are the pre-determined limits for the size of numbers. E-format is the more general form but F-format is easier to read.

In this example of E-format the E 02 part is to be understood as 10^2. E 03 would be 10^3 and E-04 would be 10^{-4}. Thus, .376 E 03 is .376 x 10^3 or 376. while .1294E-01 is .1294 x 10^{-1} or .01294. The sign following the E determines which way to move the decimal point; left for negative, right for blank or positive. The number following the sign or blank determines how many places to move the decimal point.

E. Punched Output (Don't forget to specify CARDS= on ID Card!)

All programs which have output addresses marked with the symbol Ω can punch output directly by using the X output address. X is the abbreviation for cards as output. C used as an abbreviation for an output address will be an error. Punched output generated by the use of the X output address will be in E-format. (See section above). X(F) is not a valid form and will be an error.

If punched output is desired in a form other than E-format or from a program which does not allow the X output address, then the researcher must make a copy of his data on temporary storage and go to the MATRIX program and use the PUNCH instruction provided in that program.

F. Obtaining Additional Input/Output Sources

It happens that 15 temporary storage locations may not be enough. Additional temporary storage may be obtained by calling for S16 through S40. Use of S16 through S40 requires the addition of Job Control Cards to the 360 system cards of the SOUPAC program deck. At least the first time the researcher should check with SOUPAC consultants before doing this; firstly to learn to do it correctly if he doesn't know how already, and secondly, if he knows how, to make sure none of the Job Control Language has been changed or modified, which can happen due to 360 system changes or reconfigurations, or SOUPAC system changes, which may not be announced, in contrast to SOUPAC program changes which would have been announced.

If in special instances even 40 temporary storage regions are not sufficient or a situation arises where so-called DISK temporary storage is required, there can be made available temporary storage regions called DISK1 through DISK40. Check with the SOUPAC consultants before using these for the proper Job Control Cards and the proper SOUPAC prolog cards.

G. Using Owner Data Sources or Special Input/Output Requirements in the SOUPAC system

Users' own tapes or disk packs can be used with the SOUPAC system for input or output. See Appendix A for additional information.

Special input/output requirements can usually be handled in the SOUPAC system provided the requirements can be handled by the 360 system at all. In such cases check with the SOUPAC consultants.

General problem types of the nature alluded to above would be multiple file volumes, blocked input/output, formatted or unformatted input/output, different kinds of record lengths and different forms of data representation due to machine differences or differences in facilities at other computer installations.

February, 1971

Using Owner Storage Devices With SOUPAC

Introduction

User data storage mediums for our 360 user will typically be dismountable tape or disk packs or they will be user data sets on some dismountable or permanently mounted disk pack. Whatever the source, it must be described fully so that the machine can access it. This information is needed as arguments to certain JCL keywords. These parameters will be treated in a moment. First we want to consider the dismountable devices. tapes and disk packs. User disk packs are not as frequent as user tapes so we shall ignore them for the moment. There are several "physical" characteristics of tapes which pertain to recording techniques which must be considered before we get inot the "logical" attributes described by the JCL parameters in the next section.

Physical tape parameters are density, number of tracks written and recording mode. Density can be 200, 556, 800, or 1600 bits per inch, abbreviated BPI. Thus a tape written at 800 BPI can store less information per inch than one written at 1600 BPI. Note that a tape to be used on the 360 must be written at 1600 BPI. It is true that we have a dual density tape drive, namely 800/1600 BPI accessible to the 360 and that there is a density parameter in JCL enabling one to specify the density but since the 1600 mode is so much faster, generally, to be used on the 360, the tape should be at 1600 BPI.

The number of tracks used in recording is 7 or 9. The 360 usually uses a 9 track recording mode; the 7094 used the 7 track mode. Recording mode is unformatted or formatted but we usually speak of binary tapes or card image tapes, respectively. If it is a card image tape it is BCD or EBCDIC.

Thus, the most desirable input tape for the 360 is 1600 BPI, 9 track, 360 binary or EBCDIC. If one or more of these characteristics is different, then it should be changed by doing a tape to tape copy with the appropriate conversion. Asking for special exceptions is not advised; i.e. do not try getting permission to write 800 BPI except in very special circumstances.

At this installation we can do the following conversions:

556 BPI to 800 BPI to 1600 BPI and vice versa
7 track to 9 track and vice versa
BCD to EBCDIC and vice versa

Note that in going from EBCDIC to BCD some characters may be lost. These conversions are at the moment done by the 360/50 by requesting a tape to tape copy for which the above operations are requested.

We can also do 360 binary to EBCDIC conversion and vice versa but we do not have the facility to do anything about a 7094 binary tape. Not that it cannot be done. but of the two alternative means we know of, one uses IBM conversion routines which would require hardware changes; the other requires a non-trivial program which we do not have and even when we get it, it will be slow and clumsy.

We repeat that there are JCL keywords to specify density, number of tracks, and parity but it is too slow and clumsy for the 360/75.

Clearly the greater the density, the more information can be stored on a tape and the faster it can be read or written. The higher density does however also require a better tape. One should not write 1600 BPI on a tape tested at 800 BPI. Further only a 1600 BPI tested tape can be reliably written with 9 tracks. This test BPI information is usually on the tape reel in the form of a comment saying tested at 1600 BPI or 800 BPI. Note that 1600 BPI and 3200 FCI mean the same thing.

There are several things to keep in mind when writing a user tape in binary or card image mode. Card image, line image, and formatted tape all mean essentially the same and mean exactly what their names imply. Clearly all these are read and written with a format. Binary and unformatted tapes read and write faster than formatted tapes but may take up more space than formatted tapes. How much data a user wants always accessible and how important time is will probably determine what mode to use. A card image tape may be read many different ways using formats to pick off variables. An unformatted tape has variables immediately usable but perhaps not everything the user has on his cards. He can put everything he may ever want to use on his unformatted tape but he may also be dragging some superfluous variables around everytime he does a read.

Choice, though possible, is typically not available for number of tracks on tapes written by the 360. You get 9 track tapes.

So, the usual 360 output tape is 1600 BPI, 9 track, formatted or unformatted. SOUPAC writes 1600 BPI, 9 track, unformatted tapes. To get a 1600 BPI, 9 track, formatted tape out of SOUPAC the easiest thing to do at the moment is to over-ride the punch statement in the proc, i.e., FT07F001. To input a formatted tape to SOUPAC, over-ride the procedure in an appropriate way and use the MATRIX INPUT instruction. To input a SOUPAC written unformatted tape just over-ride the procedure.

Over-riding the SOUPAC procedure for using user data storage devices

SETUP or not SETUP

Users data exists on permanently mounted volumes such as UIDCS1 or on dismountable volumes such as UIUSR4 or DK0013 or user tapes or disk packs. In this context, a volume is that physical disk pack or tape which gets mounted onto a disk drive or tape dirve. If a volume is permanently mounted you need do nothing at this setup stage. If it is dismountable, you need a /*SETUP card for each such volumne among your /* cards, if any, after the /*ID card. Tapes always require a /*SETUP card. For data sets on disk packs, determine from the user whether it is dismountable or not. The form of a /*SETUP card is as follows:

```
/*SETUP UNIT=type,ID=name
                 ,ID=(name,label).
```

Type is DISK or TAPE: the choice is obvious.

For tapes, name under ID will be ØXXXXX for an owner tape where XXXXX is some tape sequence number or PXXXXX or LXXXXX where P and L are pool and lease tapes rented from DCS.

A tape may be internally labeled or not. If it is not labeled the label argument is NL, meaning no label. If it is labelled the argument will be SL, meaning standard label. If it is standard labelled, later in the over-ride of the proc you will need to know the DSNAME.

If no label argument is used for tapes the default is SL. For dismountable disk packs at DCS, the volume name is all that is needed since they are all internally standard labelled. You must find out from a user if his tape or non-DCS disk pack is internally labelled or not.

Over-riding the SOUPAC proc

Assuming you now have the user volume mounted, you must still tell SOUPAC about it. Suppose that the user volume is a sequential data set (or possibly more than one sequential data set). Usually they will be. Direct access data sets will be discussed later. You must assign it a SOUPAC sequential data set reference number. Numbers 11 through 49 are available and correspond to the SOUPAC addresses S1 through S39 (or T1 through T39). Which data set reference number you use does not make any difference. For sake of example let us choose S 1 and over-ride FT11F001 in the SOUPAC proc. Our JCL should look like this to this point:

```
/*ID
/*SETUP UNIT=type,ID=(name,label)
//  EXEC  SOUPAC
//FT11F001  DD  indication string
//SYSIN  DD *
        .
        .
        .
```

The indication string has some or all of the parameters listed below. Use all except those where an option is mentioned.

UNIT=type	type is the same as that on the SETUP card; if no setup then it is DISK

VOL=SER=name

name is the same as that on the SETUP card; if no setup, it is the name of a permanently mounted volume

LABEL=NL
LABEL=SL

same as label argument if used on SETUP card; if not used on SETUP card, can be omitted

DISP=

arguments are NEW, OLD, KEEP, DELETE, CATALG, UNCATALG, PASS singly or in combination depending on what you are doing

DSNAME=

some name; use if you have a standard labelled tape; otherwise optional depending on what you are doing

SPACE(kind,(primary,secondary))

not necessary for an OLD, PASS, or CATALG data set; necessary for a NEW data set; not necessary for a NEW user tape data set

kind=TRK

a TRK holds 7200 bytes

CYL

a CYL has 20 tracks

there are other kind parameters, but you probably should not be using them

primary

number of kinds of space units you initially request; you get this space at once; if it is not possible to allocate you get a space unavailable message

secondary

if you run off the end of your primary allocation you get 15 extends of the number you specify as secondary of the space units you specify as kind

if the machine cannot give you 15 extents or if after 15 extents you still need space you get a space unavailable message

DCB=arg 1=arg 2
DCB=(arg 1 , arg 2 , arg 3)

optional depending on what you are doing

DCB arguments are as follows:

RECFM=	F	fixed record format (used for formatted I/O)
	V	varying record format (usually used for unformatted I/O)
	U	undefined record format (used for formatted or unformatted I/O)
	B	blocked record format
	A	alphameric record format; usually just on SYSOUT

these may be used in combination as in RECFM=FB. Note if B is used you must also use LRECL and BLKSIZE.

LRECL=n	n is the logical record length in bytes
BLKSIZE=m	m is the blocksize in bytes

If no DCB is specified you get the OS default DCB. You may not necessarily want this. Ascertain from the user what DCB he has and use it. For more information on DCB's check your FORTRAN users guide. But, we mention several pitfalls already encountered and several points of information. SOUPAC by default uses:

```
DCB=(RECFM=V,LRECL=796,BLKSIZE=800)
```

If you write a formatted data set with DISP=PASS you get RECFM=F whether you specify it or not and SOUPAC, unless you specify DCB, will kick you off with the message that ithas an F or U RECFM but expecting V. If you had a tape written with format conversion in another job and did not specify DCB=RECFM=F you will probably get an illegal decimal character message. If you give SOUPAC a data set with DCB=RECFM=F some way or another, SOUPAC will be unable to write on it since it writes unformatted unless you use the MATRIX PUNCH instruction and over-ride FT07F001. If you over-ride FT07F001 remember you get a SOUPAC DATA card image and an END# card image. If you write large formatted strings or many variables, you should check on default LRECL and BLKSIZE. If your record is larger than LRECL, you just spill onto the next logical record and the machine will worry about it. If your record is larger than BLKSIZE, you get at best that piece of the record that fits into BLKSIZE and the rest is lost. There are several tricky things that are going on here, but usually you need not worry about them. If your record is shorter than LRECL, then most of LRECL is unused and you may be wasting space. This becomes important when you start getting space unavailable or end of volume messages. Furthermore it is inefficient, though it is often much more convenient just to take the defaults. Again, consult your FORTRAN user's guide for more details.

EXAMPLES

Write onto a users tape a SOUPAC data set in binary

```
/*ID
/*SETUP  UNIT=TAPE,ID=(ØXXXXX,NL)
//  EXEC  SOUPAC
//FT12F001  DD  UNIT=TAPE,VOL=SER=ØXXXXX,LABEL=NL,DISP=(NEW,KEEP)
//SYSIN  DD  *
      .
      .
      .
TRANSFORMATIONS(IA)(T2)(13)
      .
      .
      .
```

Write onto a users tape a SOUPAC data set in EBCDIC

```
/*ID
/*SETUP  UNIT=TAPE,ID=(ØXXXXX,NL)
//  EXEC  SOUPAC
//FT07F001  DD  UNIT=TAPE,VOL=SER=ØXXXXX,LABEL=NL,DISP=(NEW,KEEP)
//SYSIN  DD  *
      .
      .
      .
MATRIX.
PUNCH(IA)"(format)"
      .
      .
      .
```

note that you will get the SYSPUNCH default DCB which limits you to card images, i.e., 80 bytes per record. For larger records use your own DCB.

Read a users formatted tape into SOUPAC

```
/*ID
/*SETUP  UNIT=TAPE,ID=ØXXXXX,SL)
//  EXEC  SOUPAC
//FT49F001  DD  UNIT=TAPE,VOL=SER=ØXXXXX,LABEL=SL,DISP=(OLD,KEEP),
              DSN=USER.PYXYY.TPDATA,DCB=(RECFM=F,LRECL=80,BLKSIZE=800)
//SYSIN  DD  *
MATRIX.
INPUT(T39)(T1)(NCOL)"(format)"
      .
      .
      .
```

note that you are reading a card image tape with 10 card images per physical record, i.e., blocking factor of 10 to 1.

Read a users unformatted tape into SOUPAC written by SOUPAC

```
/*ID
/*SETUP  UNIT=TAPE,ID=(ØXXXXX,NL)
//  EXEC  SOUPAC
//FT13F001  DD  UNIT=TAPE,VOL=SER=ØXXXXX,DISP=(OLD,KEEP),LABLE=NL
//SYSIN  DD  *
COR(T3)(P)(P).
      .
      .
      .
```

note this presupposes that SOUPAC wrote the tape originally

Read a users unformatted tape into SOUPAC not written by SOUPAC

```
/*ID
/*SETUP  UNIT=TAPE,ID=(ØXXXXX,NL)
//  EXEC  SOUPAC
//FT11F001  DD  UNIT=TAPE,VOL=SER=ØXXXXX,DISP=(OLD,KEEP),LABEL=NL,
            DCB=(RECFM=V,LRECL=400,BLKSIZE=404)
//SYSIN  DD  *
#OLD(T1)(NROW)(NCOL).
MATRIX.
MOVE(T1)(T2).
```

note first of all you need the #OLD card to simulate a SOUPAC header record. Further each record must contain three extra integer *4 words at the front of the record to correspond to SOUPAC records. Also, if it was written with default DCB you do not need to specify a DCB. Finally, the #OLD card is good to use if you are having header record problems in SOUPAC anyway.

Read a user data set from a dismountable disk

```
/*ID
/*SETUP  UNIT=DISK,ID=DK0291
//  EXEC  SOUPAC
//FT12F001  DD  UNIT=DISK,VOL=SER=DK0291,DSN=USER.PXXX.name,
            DISP=(OLD,KEEP),DCB=(RECFM=  ,LRECL=  ,BLKSIZE=  )
//SYSIN  DD  *
      .
      .
      .
```

use input if formatted; otherwise go through the appropriate sequence for unformatted reads. To write the data set, use the SPACE parameter; change the DISP parameter appropriately; and consider what kind of write you want and fix the over-ride appropriately.

If you intend to generate data in SOUPAC and wish to use the data in any context other than SOUPAC, it is highly recommended that you do not simply save the binary image file but generate a formatted record data set. Formatted record data sets interface more completely with other systems.

CORRECTION TO HANDOUT ON USING OWNER STORAGE DEVICES WITH SOUPAC

On page 5 the statements:

```
LABEL=NL
LABEL=SL
```

should read

```
LABEL=(,NL)
LABEL=(,SL)
```

therefore also in the examples every statement which has in it an occurance of the phrase LABEL=NL or LABEL=SL should have that phrase changed to LABEL=(,NL) or LABEL=(,SL), respectively.

For your information:

LABEL=(,NL) or LABEL=(,SL) is equivalent to LABEL=(1,NL) or LABEL=(1,SL) which in fact means file 1 on the volumne. Thus to access file 2 you would write LABEL=(2,NL) or LABEL=(2,SL), or in general, LABEL=(n,XX) where n is the file number and XX is NL or SL.

10/27/69

CROSS REFERENCE

Statistical Technique or Estimate	SOUPAC Source
Alpha factor analysis	ITERATIVE
Analysis of Variance	BALANOVA, T-TEST
Analysis of Covariance	T-TEST, UMAVAC*
Autocorrelations	AUTOCORRELATIONS
Beta Coefficients	CANONICAL, CORRELATIONS, MULTIPLE, STEP-WISE
Biserial Correlations	BISERIAL
Canonical Correlations	CANONICAL
Canonical Factor Analysis	ITERATIVE
Centroid Factors	CENTROID
Chi-Square	CANONICAL, CHI-SQUARE, FREQUENCY, PROBIT
Clique analysis	CLIQUE
Communality estimation	COMMUNALITIES
Correlations	AUTOCORRELATIONS, BISERIAL, CANONICAL CORRELATIONS, DISCRIMINANT, MISSING, MULTIPLE, OBLIMAX, PARTIAL, STEP-WISE
Covariance, Analysis of (See Analysis of Covariance)	
Covariance Matrix	CORRELATION, K-CLASS, MULTIPLE, STEP-WISE
Covariance Matrix of Coefficients	K-CLASS
Covariance Matrix of reduced form residuals	ECONOMETRIC
Cross-products matrix	CORRELATIONS, DISCRIMINANT, K-CLASS, MULTIPLE
Cross-Tabulation	FREQUENCY

* See SOUPAC Consultants about using UMAVAC.

Statistical Technique or Estimate	SOUPAC Source
Data Processing	MATRIX, STANDARD, TRANSFORMATIONS
Determination of a matrix	MATRIX
Determination, Coefficient of	MULTIPLE, STEPWISE, K-CLASS
Deviation covariance matrix	MULTIPLE
Deviation (partial) correlation matrix	MULTIPLE
Diagonal Method Factor Analysis	SQUARE
Dichotomous data	BISERIAL, SCALOGRAM
Discriminant Function	DISCRIMINANT
Discriminant Scores	DISCRIMINANT
Dispersion Matrix	DISCRIMINANT
Distribution Analysis	FIT, FREQUENCY, KOLMOGOROV
Durbin-Watson Coefficients	ECONOMETRIC, MULTIPLE
Dwyer Extension analysis	PROCRUSTES
Eigenvalues	DISCRIMINANT, JACOBI, K-CLASS PRINCIPAL AXIS, THREE-MODE
Eigenvectors	JACOBI, PRINCIPAL AXIS, THREE-MODE
F-Ratios	BALANOVA, DISCRIMINANT, MULTIPLE, STEP-WISE, T-TEST
Factor Analysis	COMMUNALITIES, ITERATIVE, OBLIMAX, UNRESTRICTED, VARIMAX, THREE-MODE
Factor Scores (See Factor Analysis)	
Frequency Distributions	FREQUENCY
Gamma Coefficients	FREQUENCY
Goodman and Kruskal Coefficients	FREQUENCY
Iterative Principal Axis Solution	ITERATIVE
K-Class Estimation	K-CLASS
Kolmogorov-Smirnov Statistic	KOLMOGOROV

Statistical Technique or Estimate	SOUPAC Source
Lambda Coefficients	CANONICAL, DISCRIMINANT, FREQUENCY
Latin Squares	UMAVAC
Least Squares Factor Fitting	PROCRUSTES
Limited Information Estimation	K-CLASS
Linear Programming	LINEAR
Means	CORRELATIONS, DISCRIMINANT K-CLASS, MISSING, MULTIPLE, T-TEST, STANDARD
Missing Data Correlations	MISSING
Moving Averages	STANDARD
Multiple correlation	CANONICAL, MULTIPLE, STEP-WISE
Noncard, non SOUPAC generated input	MATRIX (INPUT operation)
Oblique Factor Rotation	BINORMAMIN, OBLIMAX
Orthogonal Factor Rotation	VARIMAX
Orthogonal Simple Structure	VARIMAX
Paired Comparisons	PAIRED
Partial Correlation Coefficients	PARTIAL
Phi Coefficients	Use CORRELATIONS
Point Biserial Correlations	CORRELATIONS
Predicted Values	ECONOMETRIC, MULTIPLE
Primary Factor Pattern	BINORMAMIN, OBLIMAX
Principal Components (See Factor Analysis)	
Probit Analysis	PROBIT
Punched Card Output	MATRIX
Quadratic Programming	QUADRATIC
Random Number, Generation of	RANDOM

Statistical Technique or Estimate	SOUPAC Source
Rank Order Correlations	RANK
Recoding Variables	TRANSFORMATIONS
Reduced Form Estimates	ECONOMETRIC
Reduced Form Predicted Values	ECONOMETRIC
Reduced Form Residuals	ECONOMETRIC
Reference Vector Structure	OBLIMAX, BINORMAMIN
Regression Analysis	CORRELATION, K-CLASS, MULTIPLE, STEP-WISE
Regression Covariance	MULTIPLE
Residual Analysis	ECONOMETRIC
Rotation of Factors	BINORMAMIN, OBLIMAX VARIMAX
Sample Size	CORRELATION, DISCRIMINANT, FREQUENCY, MISSING, MULTIPLE, STANDARD
Serial Correlations	AUTOCORRELATIONS
Spearman's Rho	RANK
Spearman's C	FREQUENCY
Spectrum Analysis	AUTOCORRELATIONS
Square Root Factors	SQUARE
Standard Deviations	CORRELATION, MULTIPLE, STANDARD, T-TEST
Standard Errors of Regression Coefficients	K-CLASS, MULTIPLE, STEP-WISE
Standardized Data	STANDARD SCORES
Standardized Regression Coefficients	CANONICAL, MULTIPLE, STEP-WISE
Stepwise Maximum Likelihood Factor Analysis	ITERATIVE
Stepwise Multiple Regression	STEP-WISE
T-Statistic (Student's)	CORRELATION, MULTIPLE, STEP-WISE T-TEST
Tchuprow's T	FREQUENCY

Statistical Technique or Estimate	SOUPAC Source
Tetrachoric Correlations	PARTIAL
Three Mode Factor Analysis	THREE MODE
Three Stage Least Squares	THREE STAGE
Two Stage Least Squares	K-CLASS
Unrestricted Maximum Likelihood Factor Analysis	UNRESTRICTED
Variance (See Standard Deviations)	
Variance, Analysis of (See Analysis of Variance)	

DATA AND MATRIX MANIPULATIONS PACKAGE

The data manipulative programs

As implied by the name, TRANSFORMATIONS is used for performing data transformations often necessary in "setting up" data to be input to one of the "cook-book" programs of the SOUPAC system. The user may create new variables as linear combinations or as algebraic functions of old variables, and the user may recode or alter data values on the basis of test conditions by using the TRANSFORMATIONS program. To perform matrix algebra operations with one or more matrices, to augment matrices either row-wise or column-wise, to reorder, save or delete specific rows or columns of a matrix, one would use the MATRIX program. MATRIX also has the capability of printing, punching card decks, and reading and writing tape and disk files in ways not available elsewhere in SOUPAC.

The TRANSFORMATIONS and MATRIX programs have a unique place in the SOUPAC system. Although the remainder of the SOUPAC library performs a large number of specialized statistical procedures, there are some computations which are not represented by a uniquely written program. However, by an imaginative utilization of the combined powers of TRANSFORMATIONS and MATRIX it is possible to perform a virtually unlimited range of established and experimental statistical techniques. It is a common practice within the SOUPAC office to "check out" newly written programs against results computed by TRANSFORMATIONS and MATRIX. Conversely, these two programs can be used as teaching tools by having students learn the step-by-step computations and then checking the results against results of the "cook-book" programs. Complete multiple regressions and analysis of variance programs, for example, have been written in this manner. Since most SOUPAC jobs require the use of TRANSFORMATIONS and MATRIX, a familiarity with these two programs is basic to an effective use of the SOUPAC system.

MATRIX

I. General Description

The MATRIX program is a data manipulating program for inputting and outputting, creating, performing matrix algebraic operations, and generally handling data matrices. All the MATRIX suboperations are restricted to 450 columns (variables). No absolute limit is set on the number of rows (observations).

Standard SOUPAC address conventions are used including the use of the character X to denote punched output, and (F) after a print to denote print with F format. Also available and discussed in section III below is the use of I for storing a matrix in memory, and the use of (L) after a print to invoke the MATRIX labeling feature. All other restrictions are noted by the discussion of the individual suboperation explanations.

II. Parameters

A. Main Parameters

To invoke the MATRIX program, code the name MATRIX (or simply the program mnemonic MAT). There are two optional parameters available which may be coded on the MAT card.

First, if it is desired to print, immediately prior to the execution of each MATRIX subparameter operation, the time in seconds since entry into the MATRIX program, code a (1) after the name MATRIX. This option is not normally needed and is provided merely for giving timing estimates.

The second optional parameter is coded as a (1) following the timing estimate parameter. This second option causes the number of rows and columns and the precision (either single or double)of the answer matrix (for all suboperations which produce an answer matrix) to be printed out. This is useful in debugging a MATRIX program when it is not necessary to see the entire answer matrix to be printed out, but it would be helpful to check the dimensions of an answer matrix. As in all SOUPAC programs, the main program parameter card must be terminated by a period. Examples:

```
MATRIX.

MATRIX (1).

MAT.

MAT (1).

MATRIX (1)(1).

MAT ( )(1).  -- recommended usage --
```

B. Subparameters

Any MATRIX operation may be invoked by coding its mnemonic followed by appropriate subparameters. All operations in MATRIX handle both single and double precision matrices at the control of the user (see operations SINGLE and DOUBLE). For an address not explicitly assigned either single or double precision, MATRIX assumes a default of double precision for output to the address. Terminate all subparameter cards with a period.

To end a MATRIX program, place a card which has the characters END P after the last MATRIX subparameter card. Since all MATRIX programs must have at least one subparameter operation, an error will be signaled if a MAT card is followed immediately by an END P card.

Input and output for MATRIX may be from any source, however, the following rules must be observed:

1) Never use CARDS as input to any operation except MOVE unless both the number of rows and the number of columns have been specified on the DATA format card at the front of the data deck.

2) You may never output to PRINT only. All MATRIX output must go to some intermediate storage location even when only printout is desired.

3) Avoid using the same address more than once on the same parameter card unless otherwise noted in the description of an individual operation. However, in those operations which do permit using an address more than once as an input address, CARDS <u>may not</u> be used as an input address more than once. In all operations except INVERT, <u>never</u> specify an output address which is the same as an input address for that operation.

4) The contents of an input address remains unchanged during the execution of an operation unless otherwise noted.

Following is a description of the subparameter operations currently in the MATRIX program.

Summary of MATRIX Operations

mnemonic	notes	operation name	examples
ABS		Absolute value	ABS (S1) (S2).
ADD	1,2,5	Add	ADD (S1) (S2) (S3).
ALL	3,6	All	ALL (S1) "GT" *0* (S2). ALL (S1) "NE" *-0* (S2) (2). ALL (S1) "LE" (S2) (S3) (1,10,3).
ANY	3,6	Any	ANY (S1) "GE" (S2) (S3). ANY (S1) "EQ" *1.* (S2) (1,2) (5). ANY (S1) "LT" *99* (S2).
CHO	4	Cholesky decomposition	CHO (S1) (S2).
COL	6	Column delete	COL (S1) (S2) (2). COL (S1) (S2) (10) (12,15).
CON		Constant addition	CON (S1) *2.* (S2). CON (S1) (S2) (S3).
COU		Count	COU (S1) (S2). COU (S1) (S2) (1). COU (S1) (S2) (2).
DIA		Diagonal to vector	DIA (S1) (S2).
DIM		Dimension	DIM (S1) (S2).
DOU	2	Double precision	DOU (S1).
EJE		Eject	EJE.
E-D	1,2,5	Elementwise divide	E-D (S1) (S2) (S3).
E-M	1,2,5	Elementwise multiply	E-M (S1) (S2) (S3).
E-R		Elementwise square root	E-R (S1) (S2).
EXP		Expand	EXP (S1) *20* (S2). EXP (S1) (S2) (S3).
FIL	2	File	FIL (S1).
GEN	6	Generate	GEN (S1) *1*.
HOR	1,2,5	Horizontal augment	HOR (S1) (S2) (S3).
IDE		Identity matrix	IDE (48) (S1).
INP		Input	INP (S1) (S2) ( ) (9) "(9F6.1)". INP (S1) (S2) ( ) (12). INP (S1) (I) (576) (10).
INV	4	Invert	INV (S1) (S2). INV (I) (I). INV (S1) (S2) (1) (1) *10.E-6*.
LAB	6	Label	LAB (S1) "SAMPLE 1" "AGE" "SEX".
LAG		Lag	LAG (S1) (2) (6) ( ) (S2). LAG (S1) (3) (4) (1) (S2).
LOW		Lower triangle	LOW (S1) (S2). LOW (S1) (S2) (1).

mnemonic	notes	operation name	examples
MAX		Maximum value	MAX (S1) (S2). MAX (S1) (S2) (1). MAX (S1) (S2) (2).
MIN		Minimum value	MIN (S1) (S2). MIN (S1) (S2) (1). MIN (S1) (S2) (2).
MOV	2	Move	MOV (C) (S1). MOV (S1) (S2).
MUL	1,4,5	Multiply	MUL (S1) (S2) (S3).
OUT		Output	OUT (S1) (S2) "(10E16.9)". OUT (S1) (S2).
PAR		Partition	PAR (S1) (S2) (5) (10) (2) (21).
PER	6	Permutation	PER (S1) (2) (1).
PRI		Print	PRI (S1) "(' ',10F13.4)".
PUN		Punch	PUN (S1) "(8F10.3)".
REC		Reciprocal	REC (S1) (S2).
REM		Remap	REM (S1) (S2) (8).
REW	2	Rewind	REW (S1).
ROW	6	Row delete	ROW (S1) (S2) (1) (2) (3). ROW (S1) (S2) (2,20,3) (3,20,3).
SCA		Scalar multiply	SCA (S1) *.1* (S2). SCA (S1) (S2) (S3).
SIN	2	Single precision	SIN (S1).
SUB	1,2,5	Subtract	SUB (S1) (S2) (S3).
SUM		Sum	SUM (S1) (S2). SUM (S1) (S2) (1). SUM (S1) (S2) (2).
TRA	4	Transpose	TRA (S1) (S2).
UPP		Upper triangle	UPP (S1) (S2). UPP (S1) (S2) (1).
VEC		Vector to diagonal	VEC (S1) (S2).
VER	1,2,5	Vertical augment	VER (S1) (S2) (S3).

1. Conformability of input matrices is checked.
2. Up to twenty-one total addresses may be used.
3. A warning message is printed if no rows are output.
4. Any matrix which has been previously stored under the I address will be destroyed.
5. An input address may be used more than once for input to the same instruction.
6. As many arguments as are needed of the last argument type may be used.

ABSOLUTE VALUE (mnemonic: ABS)

The ABSOLUTE VALUE operation has two address parameters, an input address and an output address. The absolute value of each element of the input matrix is taken and the result goes to the output address. Example:

```
ABSOLUTE VALUE (SEQ3)(SEQ4).
```

ADD (mnemonic: ADD)

The ADD operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows and columns as all other input matrices. An address may be used more than once as an input address.

Corresponding elements of the first matrix through the next to last matrix are added together, and the result goes to the output address. Examples:

```
ADD (SEQ1)(SEQ4)(SEQ5).
ADD (SEQ1)(SEQ4)(SEQ2)(SEQ3)(SEQ5).
```

ALL (mnemonic: ALL)

The ALL operation performs a particular test, specified by the second operand as a relational operator, between a set of elements for each input row and a floating point number specified by the third operand. If all elements of the set for a given row pass the test, that row is output to the output address.

The first parameter is the input address. The relational operator is enclosed in quotation marks. The third operand may be either a floating point number or an address in which case the first element of the matrix is used as the floating point number. The six legal relational operators are "LT", "LE", "EQ", "NE", "GT", and "GE". Remaining (optional) parameters are index sets specifying which variables are to be included in the testing. If no variables are specified, all variables are included in the testing. Note that if only one variable is specified, the results of the ANY and the ALL operation would be the same. Examples:

```
ALL (SEQ1) "NE" *0.* (SEQ2).
ALL (SEQ3) "GE" (SEQ4)(SEQ1).
```

ANY (mnemonic: ANY)

The ANY operation performs a particular test, specified by the second operand as a relational operator, between a set of elements for each input row and a floating point number specified by the third operand. If any element of the set for a given row passes the test, that row is output to the output address.

The first parameter is the input address. The relational operator is enclosed in quotation marks. The third operand may be either a floating point number or an address in which case the first element of the matrix is used as the floating point number. The six legal relational operators are "LT", "LE", "EQ", "NE", "GT", and "GE". Remaining (optional) parameters are index sets specifying which variables are to be included in the testing. If no variables are specified, all variables are included in the testing. Note that if only one variable is specified, the results of the ANY and the ALL operation would be the same. Examples:

```
ANY (SEQ2) "GT" *3.* (SEQ1).
ANY (SEQ1) "NE" *-0.* (SEQ4)(1,3).
```

CHOLESKY (mnemonic: CHO)

The CHOLESKY operation decomposes a square symmetric matrix into the product of an upper triangular matrix and a lower triangular matrix such that the two triangular matrices are the transpose of each other. CHOLESKY has two operands, an input address and an output address. The result which goes to the output address is the lower triangular matrix resulting from the decomposition.

If the input matrix is not square, the "extra" rows or columns are ignored. Additionally, if the square matrix is not symmetric, the actual upper triangle of the matrix is effectively ignored and is instead assumed to be identical to the lower tirangle. Example:

```
CHOLESKY (SEQ1)(SEQ2).
```

If the input matrix on SEQ 1 is

```
 4    -2    -4
-2     2     3
-4     3     6
```

the resulting matrix output to SEQ 2 is

```
 2     0     0
-1     1     0
-2     1     1
```

COLUMN DELETE (mnemonic: COL)

The COLUMN DELETE operation specifies which columns of an input matrix are to be deleted before sending the result to the output address. The first parameter is the input address. The second parameter is the output address. Columns to be deleted are specified by index sets following the output address. Examples:

```
COLUMN DELETE (SEQ4)(SEQ5)(4)(5)(6)(8)(13)(15).
COLUMN DELETE (SEQ1)(SEQ2)(1,30,3).
```

CONSTANT (mnemonic: CON)

The CONSTANT operation has three parameters. A floating point number specified by the second operand is added to every element of the matrix specified by the first operand. The result goes to the third operand address.

The second operand may be either a floating point number enclosed in asterisks or a standard SOUPAC input address. If an address is specified, the first element of the matrix at the address is used for the floating point number. Examples:

```
CONSTANT (SEQ1) *4.5* (SEQ2).
CONSTANT (SEQ1)(SEQ3)(SEQ4).
```

COUNT (mnemonic: COU)

The COUNT operation has three operands; an input address, an output address, and an option indicator.

If option 0 is specified, the resulting output matrix is a single row vector containing a count of the number of elements, excluding missing data, of each column of the input matrix. Specifying no option is equivalent to specifying option 0.

If option 1 is specified, the resulting output matrix is a single column vector containing a count of the number of elements, excluding missing data, of each row of the input matrix.

If the option is specified as any number other than 0 or 1, a single element matrix is output which contains a count of the number of elements, excluding missing data over the entire matrix. Examples:

```
COUNT (SEQ1)(SEQ4).
COUNT (SEQ5)(SEQ3)(1).
COUNT (SEQ2)(SEQ3)(2).
```

DIAGONAL (mnemonic: DIA)

The DIAGONAL operation has two operands, an input address and an output address. The main diagonal elements of the first matrix are used to form a single row vector which is output to the second operand address. Example:

```
DIAGONAL (SEQ2)(SEQ4).
```

DIMENSION (mnemonic: DIM)

The DIMENSION operation has two address parameters, an input address, and an output address. The number of rows and the number of columns of the input matrix are used to form the first and second elements respectively of a two element, single row matrix which is output to the output address. Example:

```
DIM (SEQ2)(SEQ1).
```

DOUBLE (mnemonic: DOU)

The DOUBLE operation has anywhere from one to twenty-one addresses as parameters. Listing an address as a parameter negates the effect of any previous listing of that address as a parameter in the operation SINGLE. Listing an address as a parameter which has not appeared as a SINGLE subparameter has no effect. Example:

```
DOUBLE (SEQ1)(SEQ2)(SEQ3)(SEQ4)(SEQ5)(I).
```

EJECT (mnemonic: EJE)

The EJECT operation causes the next printout to begin at the top of a new page. EJECT has no parameters. Example:

```
EJECT.
```

E-DIVIDE -- Elementwise Divide -- (mnemonic: E-D)

The E-DIVIDE operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows and columns as all other input matrices for the use of the operation. An address may be used more than once as an input address.

Elements of the second matrix through the next to last matrix are divided into the corresponding elements of the first matrix. Output goes to the last address. Example:

```
E-DIVIDE (SEQ1)(SEQ2)(SEQ3).
```

E-MULTIPLY -- Elementwise Multiply -- (mnemonic: E-M)

The E-MULTIPLY operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows and columns as all other input matrices for the use of the operation. An address may be used more than once as an input address.

Corresponding elements of the first matrix through the next to last matrix are multiplied together. Output goes to the last address. Example:

```
E-MULTIPLY (SEQ3)(SEQ2)(I)(SEQ4).
```

E-ROOT -- Elementwise Square Root -- (mnemonic: E-R)

The E-ROOT operation has two address parameters, an input address and an output address. The (positive) square root of each element of the input matrix is taken and the result goes to the output address. Example:

```
E-ROOT (SEQ1)(SEQ2).
```

EXPAND (mnemonic: EXP)

The EXPAND operation takes the first row of the first input matrix and repeatedly outputs that same row to the output address the number of times specified by the second parameter.
The second parameter can be either a floating point number in which case the input row is copied to the output address the number of times specified by the integer portion of the floating number; or the second parameter can be an input address in which case the input row is copied to the output address until the output matrix has the same number of rows as the second input matrix. The third parameter is the output address. Examples:

```
EXPAND (SEQ1)(SEQ2)(SEQ3).
EXPAND (SEQ1)*55*(SEQ4).
```

FILE (mnemonic: FIL)

The FILE operation has anywhere from one to twenty-one addresses as parameters. FILE is used to cause an end-of-file mark to be written at the end of a SEQUENTIAL file. This operation is generally most useful to the user who wishes to place more than one file on his own physical tape. Since any meaningful use of the FILE operation requires the addition of appropriate IBM 360 JCL cards, all but the most experienced users should see a consultant in the SOUPAC office before using this operation. Example:

```
FILE (SEQ5).
```

GENERATE (mnemonic: GEN)

The GENERATE operation generates a single row vector with the floating point numbers the user specifies. The first operand is the output address. Remaining parameters are as many floating point numbers as the user wishes. Example:

```
GENERATE (SEQ2) *1.* *2.* *4.* *8.* *16.* *32.*.
```

HORIZONTAL AUGMENT (mnemonic: HOR)

The HORIZONTAL AUGMENT operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows as all other input matrices for the use of the operation. An address may be used more than once as an input address.
Input matrices from the first matrix through the next to last matrix are stacked left to right and the result goes to the last address. Example:

```
HORIZONTAL AUGMENT (SEQ1)(SEQ4)(SEQ2)(I).
```

IDENTITY (mnemonic: IDE)

The IDENTITY operation has two parameters. An identity matrix, of order specified by a fixed point number as the first operand, is output to the address specified by the second operand. Example:

```
IDENTITY (20)(SEQ3).
```

INPUT (mnemonic: INP)

The INPUT operation will input formatted or non-formatted records from any available device. This option is primarily for reading card images or other similar data the user may have usually on his own tape, which would be awkward to input in the typical card deck manner.

Never input to I (see SPECIAL COMMENTS) using the INPUT operation unless both the number of rows and the number of columns of the input matrix are specified as parameters on the INPUT operation parameter card.

The parameters for INPUT are the input address, the output address, the number of rows of the input matrix (optional in most cases), number of columns, and optionally the format enclosed in quotation marks. Examples:

```
INPUT (SEQ1)(SEQ2/PRINT)(20)(5) "(10F8.3)".
INPUT (SEQ2)(SEQ3)( )(8).
```

INVERT (mnemonic: INV)

The INVERT operation inverts a non-singular real matrix. The INVERT operation has five subparameters, the last three of which are optional. The first parameter is the address of the matrix to be inverted, and the second parameter is the output address of the result. (The incore address option described in section III.A - SPECIAL COMMENTS - may be used for either input, output, or both). To have the determinant of the original matrix printed out, code a (1) as the third parameter.

The inversion technique used is the Gauss-Jordan method with pivot elements assumed to be on the main diagonal. If it is desired that the inversion technique perform row and column interchange, for the purpose of picking pivot elements as those with the largest absolute value at each step of the elimination procedure, code a (1) as the fourth parameter. The default case, pivot elements assumed to be on the main diagonal, executes faster than when row and column interchange is performed. For those real symmetric matrices which have the property that the largest elements are necessarily on the main diagonal (e.g. correlation, cross-products, variance-covariance matrices) numerical accuracy of the results is not significantly different between the two options. For general matrices in which specific properties are not known, using row and column interchange will probably produce more accurate results.

The fifth argument is a floating point number enclosed in asterisks which is to be used as the criterion for singularity. If the absolute value of any pivot element is less than the criterion for singularity, the matrix is assumed to be singular. If no value is specified, or if *0.* is specified as the fifth parameter, a default value of 10^{-8} is used to test for singularity.

INVERT destroys any previous use of the incore address option. All calculations are done in double precision.

INVERT also has the ability to solve a set of simultaneous linear equations if a unique solution exists. To solve the system indicated by the matrix equation

$$AX = Y$$

input to the INVERT suboperation a matrix which contains A:Y (i.e. the constant term appearing as the last column variables). The resulting output of the INVERT suboperation will be

$$A^{-1}:X$$

The Y above may be more than one column vector in which case each resulting column vector of X will be the solution for the corresponding column of Y. Examples:

```
INVERT (SEQ1)(SEQ2).
INVERT (SEQ4)(SEQ3)(1)(1).
INVERT (SEQ2)(SEQ4)(1)( ) *10.E-5*.
INVERT (SEQ5)(SEQ1)( )(1) *.0000001*.
```

<u>LABEL</u> (mnemonic: LAB)

The LABEL operation is used to store a title and column labels at a SOUPAC address for later use within a MATRIX program. The title is limited to 128 characters. Labels are limited to eight characters each.

The first parameter is the address where the title and labels are to be stored. This is then followed by the title and labels each enclosed in quotation marks. Only one label set is active at any one time. Hence, each use of LABEL overrides all previous uses. Labels generated within a MATRIX program may not be passed to other programs. (note: The incore address option may be used to store a title and label set if desired.) Examples:

```
LABEL (SEQ4) "TITLE" "LABEL 1" "LABEL 2" "LABEL 3".
LABEL (SEQ1) "ONLY A TITLE; NO LABELS".
```

<u>LAG</u> (mnemonic: LAG)

The LAG operation has five operands; an input address, an integer specifying which variable is to be lagged, an integer specifying the number of lag periods, an integer option flag, and an output address.

If the option flag is zero, intermediate lags are included in the output matrix; if the option flag is non-zero, intermediate lags are not included.

For example, suppose we are interested in lagging the fourth variable of a five variable matrix and suppose we want three lag periods. First it should be noted that the resulting output matrix will necessarily have three fewer rows (observations) than the input matrix.

If we use

```
LAG(S1)(4)(3)(0)(S2).
```

the resulting tth row to the output address would be

$$X_{t,1}\ X_{t,2}\ X_{t,3}\ X_{t,4}\ X_{t,5}\ X_{t-3,4}\ X_{t-2,4}\ X_{t-1,4}$$

If we use

```
LAG(S1)(4)(3)(1)(S2).
```

the resulting tth row to the output address would be

$$X_{t,1}\ X_{t,2}\ X_{t,3}\ X_{t,5}\ X_{t-3,4}$$

Note that the difference between the option flag being zero and non-zero is that the resulting output matrix has N-1 fewer lags where N is the number of lag periods (i.e. the intermediate lags are not included in the output).

As a concrete example consider the following input on S1.

```
 1     2     3
 4     5     6
 7     8     9
10    11    12
13    14    15
16    17    18
```

Use of the statement

```
LAG(S1)(2)(2)( )(S2).
```

will result in the following matrix to be output to S2.

```
 7     8     9     2     5
10    11    12     5     8
13    14    15     8    11
16    17    18    11    14
```

Use of the statement

```
LAG(S1)(2)(2)(1)(S2).
```

will result in the following matrix to be output to S2.

```
 7    8    9    2
10   11   12    5
13   14   15    8
16   17   18   11
```

LOWER TRIANGLE (mnemonic: LOW)

The LOWER TRIANGLE instruction copies a matrix from one address to another and sets all elements which are above the main diagonal to zero. It is possible to indicate if it is desired that the main diagonal elements also be set to zero.

The LOWER TRIANGLE instruction has three operands; an input address, an output address, and an integer option flag. If the option flag is omitted or is zero, the main diagonal elements are included as part of the lower triangle. If the option flag is non-zero, the main diagonal elements are set to zero. Examples:

```
LOWER (S1)(S3).
LOWER (S2)(S4)(1).
```

MAXIMUM (mnemonic: MAX)

The MAXIMUM operation has three operands; an input address, an output address, and an option indicator.

If option 0 is specified, the resulting output matrix is a single row vector containing the maximum element of each column of the input matrix. Specifying no option is equivalent to specifying option 0.

If option 1 is specified, the resulting output matrix is a single column vector containing the maximum element of each row of the input matrix.

If the option is specified as any number other than 0 or 1, a single element matrix is output which contains the maximum element of the entire matrix. Examples:

```
MAX (SEQ1)(SEQ3).
MAX (SEQ1)(SEQ2)(2).
```

MINIMUM (mnemonic: MIN)

The MINIMUM operation has three operands; an input address, an output address, and an option indicator.

If option 0 is specified, the resulting output matrix is a single row vector containing the minimum element of each column of the input matrix. Specifying no option is equivalent to specifying option 0.

If option 1 is specified, the resulting output matrix is a single column vector containing the minimum element of each row of the input matrix. Examples:

```
MIN (SEQ1)(SEQ3).
MIN (SEQ1)(SEQ2)(2).
```

MOVE (mnemonic: MOV)

The MOVE operation moves (actually copies) a matrix from one SOUPAC standard input source to another. If reading from SEQUENTIAL, the MOVE operation assumes that the data set was created using SOUPAC conventions, i.e. by some SOUPAC program. If the input source is CARDS, the input deck must be preceded by a correct DATA format statement and terminated by an END# card.

Never MOVE from CARDS to I (see SPECIAL COMMENTS) unless both number of rows and number of columns of the input matrix are specified at the front of the data deck.

The operation has between two and twenty-one addresses as parameters. The first address is the input address. All remaining addresses are output addresses. Examples:

```
        MOVE (CARDS)(SEQ1)(SEQ2).
        MOVE (CARDS)(SEQ1).
```

MULTIPLY (mnemonic: MUL)

The MULTIPLY operation has three addresses for parameters. A matrix multiplication is performed between the matrices on the first two addresses and the result is stored in the third address. The MULTIPLY operation permits use of the same address to be used as an input address for both first and second operands. This usage is equivalent to using the SQUARE suboperation.

The incore address option may not be used for input of the first operand if the first operand is different from the second. The incore address option may never be used for output of the result. The MULTIPLY operation destroys any matrix which has been stored in core using the incore address option (see section III.A - SPECIAL COMMENTS). All calculations are done in double precision. Examples:

```
        MULTIPLY (SEQ1)(SEQ2)(SEQ3).
        MULTIPLY (SEQ1)(SEQ1)(SEQ2).
        MULTIPLY (SEQ2)(I)(SEQ5).
```

OUTPUT (mnemonic: OUT)

The OUTPUT operation outputs a matrix, a row at a time with or without format control, to a user specified data set. If format control is used, the output address specified should not be used anywhere else in the current SOUPAC job step except with options which also perform formatted I/O (e.g. the INPUT and OUTPUT operations of MATRIX). The syntax of OUTPUT is two addresses, input and output addresses respectively, optionally followed by the desired format enclosed by quotation marks. Examples:

```
        OUTPUT (SEQ1)(SEQ5) "(20E15.7)".
        OUTPUT (SEQ2)(SEQ3).
```

PARTITION (mnemonic: PAR)

The PARTITION operation is used to select a sub-matrix of an original input matrix. The first operand is the input address and the second operand is the output address. The next four parameters specify in order, the beginning column of the partition, the ending column of the partition, the beginning row of the partition, and the ending row of the partition. If either beginning parameter is left out, the partition begins with the first row (or column). If either ending parameter is left, the partition ends with the last row (column). Examples:

```
PARTITION (SEQ5)(SEQ2)(5)(6)(2)(50)
PARTITION (SEQ4)(SEQ2)(3)(40).
```

PERMUTATION (mnemonic: PER)

The PERMUTATION operation permutes, on option, rows or columns or rows and columns of an input matrix. The resulting matrix is output to the second operand output address.

The third operand is an option flag. If the option flag is specified as zero, columns of the matrix are permuted. If the option flag is specified as one, rows of the matrix are permuted. If the option flag is specified as other than zero or one, both rows and columns are permuted.

The order of columns or rows of the output matrix is determined by index sets. Examples:

If the statement

```
PERMUTE (S1)(S2)(0)(5)(1,4).
```

is used and S1 has the matrix

```
 1.    2.    3.    4.    5.
-1.    0.    2.     .1    .3
```

the resulting output matrix will be

```
 5.    1.    2.    3.    4.
  .3  -1.    0.    2.    1.
```

If the input matrix had been

```
 1.    2.    3.    4.    5.    6.
-1.    0.    2.     .1    .3   4.
```

the resulting output matrix would be the same as the output matrix already listed. Note that this implies that only those columns or rows which are explicitly listed will be output.

If the statement

```
PER (S1)(S2)(1)(3)(2)(1).
```

is used and the matrix on S1 is

```
1    2    3
4    5    6
7    8    9
```

the resulting output matrix will be

```
3    2    1
6    5    4
9    8    7
```

If the statement

```
PER (S1)(S2)(2)(3)(2)(1).
```

is used on the matrix

```
1    2    3
4    5    6
7    8    9
```

the resulting output matrix will be

```
9    8    7
6    5    4
3    2    1
```

PRINT (mnemonic: PRI)

The PRINT operation prints out a matrix, one row at a time, under the control of a user supplied format. Formats follow FORTRAN IV conventions with the added restriction that formats are limited to 592 characters.

The first parameter is the address of the matrix to be printed. The second parameter is the format enclosed in quotation marks. (Warning: Allow for carriage control as the first character in output lines. A print line has 133 characters.) Example:

```
PRINT (SEQ2) "(' ',3F20.10)".
```

PUNCH (mnemonic: PUN)

The PUNCH operation has the same syntax as PRINT and is used to punch out a matrix under the control of a user supplied format. (Warning: When punching cards, remember that there is room for only 80 characters per card).

The PUNCH operation always punches two cards in addition to the actual data deck. At the front of the data is punched a DATA format card, and at the end of the data is punched an END# card. Example:

```
PUNCH (SEQ1)"(8F10.2)".
```

RECIPROCAL (mnemonic: REC)

The RECIPROCAL operation has two operands, an input address and an output address. The reciprocal of the elements from the first matrix are used to form an output matrix which is output to the second operand address. Example:

```
RECIPROCAL (SEQ1)(SEQ3).
```

REMAP (mnemonic: REM)

The REMAP instruction is used when a row of the input matrix represents several observations and it is desired to "break up" the input row into several shorter output rows.

The REMAP instruction has three operands; an input address, an output address, and an integer which is to be used as the column dimension of the output address.

For example, if we have the row of data on S1

```
1.    2.    3.    4.    5.    6.
```

and use the instruction

```
REMAP (S1)(S2)(2).
```

the resulting output to S2 would be

```
1.    2.
3.    4.
5.    6.
```

Notice that the column dimension to be specified for the output matrix must exactly divide the column dimension of the input matrix.

REWIND (mnemonic: REW)

The REWIND operation has anywhere from one to twenty-one addresses as parameters. REWIND is used to rewind a sequential file. The REWIND operation needs only to be used with the INPUT operation when it is desired to reread a formatted input file. Example (to input the same formatted file from SEQ 3 onto both SEQ 1 and SEQ 2 under control of different formats):

```
INPUT (SEQ3)(SEQ1)( )(5) "(10X,5F10.0)".
REWIND (SEQ3).
INPUT (SEQ3)(SEQ2)( )(8) "(8F10.0)".
```

ROW DELETE (mnemonic: ROW)

The ROW DELETE operation specifies which rows of an input matrix are to be deleted before sending the result to the output address. The first parameter is the input address. The second parameter is the output address. Rows to be deleted are specified by index sets following the output address. Example:

```
ROW DELETE (I)(SEQ3)(1)(3)(7)(8)(11)(45).
```

SCALAR (mnemonic: SCA)

The SCALAR operation has three parameters. A floating point number specified by the second operand is multiplied by every element of the matrix specified by the first operand. The result goes to the third operand address.

The second operand may be either a floating point number enclosed in asterisks or a standard SOUPAC input address. If an address is specified, the first element of the matrix at the address is used for the floating point number. Examples:

```
SCALAR (SEQ1) *2.* (SEQ2).
SCALAR (SEQ4)(SEQ3)(SEQ2).
```

SINGLE (mnemonic: SIN)

The SINGLE operation has anywhere from one to twenty-one addresses as parameters. Listing an address as a parameter causes any matrices written on that address to be written in single precision. MATRIX stores all data matrices in double precision unless the user specifies otherwise with the suboperation SINGLE. The listing of an address in a SINGLE statement in one MATRIX program does not carry over in effect to any other MATRIX program. Example:

```
SINGLE (SEQ1)(SEQ2)(SEQ4).
```

SUBTRACT (mnemonic: SUB)

The SUBTRACT operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of rows and columns as all other input matrices for the use of the operation.
Elements of the second matrix through the next to last matrix are subtracted from corresponding elements of the first matrix. Output goes to the last address. An address may be used more than once as an input address. Examples:

```
SUBTRACT (SEQ1)(SEQ3)(SEQ4).
SUBTRACT (SEQ4)(SEQ2)(SEQ3)(SEQ1)(SEQ5).
```

SUM (mnemonic: SUM)

The SUM operation has three operands; an input address, an output address, and an option indicator.
If option 0 is specified, the resulting output matrix is a single row vector containing the column sum of each column of the input matrix. Specifying no option is equivalent to specifying option 0.
If option 1 is specified, the resulting output matrix is a single column vector containing the row sum of each row of the input matrix.
If the option is specified as any number other than 0 or 1, a single element matrix is output which contains the sum of all elements over the entire matrix. Examples:

```
SUM (SEQ1)(SEQ3).
SUM (SEQ1)(SEQ2)(2).
```

TRANSPOSE (mnemonic: TRA)

The TRANSPOSE operation transposes a matrix (interchanges rows and columns). TRANSPOSE destroys any previous usage of the incore address storage. The two parameters for TRANSPOSE are first the input address and second the output address of the result. Example:

```
TRANSPOSE (SEQ1)(SEQ2).
```

UPPER TRIANGLE (mnemonic: UPP)

The UPPER TRIANGLE instruction copies a matrix from one address to another and sets all elements which are below the main diagonal to zero. It is possible to indicate if it is desired that the main diagonal elements also be set to zero.
The UPPER TRIANGLE instruction has three operands; an input address, an output address, and an integer option flag. If the option flag is omitted or is zero, the main diagonal elements are included as part of the upper triangle. If the option flag is non-zero, the main diagonal elements are set to zero. Examples:

```
UPPER (S1)(S3).
UPPER (S2)(S4)(1).
```

VECTOR (mnemonic: VEC)

The VECTOR operation has two operands, an input address and an output address. A single vector from the first location is used to form a diagonal matrix which is output to the second address. If the input matrix has more rows than columns, the first column vector is used to form the diagonal matrix. If the input matrix has more columns than rows, the first row is used to form the diagonal matrix. Example:

```
VECTOR (SEQ1)(SEQ3).
```

VERTICAL AUGMENT (mnemonic: VER)

The VERTICAL AUGMENT operation has from three to twenty-one address parameters. The last address is the output address; all other addresses are for input. Each input matrix must have the same number of columns as all other input matrices for the use of the operation. An address may be used more than once as an input address.

Input matrices from the first address through the next to the last address are stacked top to bottom and the result goes to the last address. All input matrices must have the same number of columns. Example:

```
VERTICAL AUGMENT (SEQ1)(SEQ2)(SEQ3).
```

Addendum to the MATRIX Program Description

KRONECKER PRODUCT (mnemonic: KRO)

The KRONECKER PRODUCT operation forms the Kronecker Product of two matrices and outputs the results to an output address. The resulting output matrix is composed of m_1 x n_1 submatrices where m_1 and n_1 are the dimensions of the first input matrix. Each submatrix has the size m_2 x n_2 where m_2 and n_2 are the dimensions of the second input matrix. Note that the output matrix has the dimensions m_1m_2 x n_1n_2. Each submatrix is the result of the scalar product a_{ij} B.

For example, if matrix A is on S1 and B is on S2 where

$$A= \begin{array}{cc} 1. & 2 \\ -1 & 1 \\ 0 & .5 \end{array} \qquad B= \begin{array}{cc} 1 & 3 \\ 2 & 4 \end{array}$$

the result of executing the MATRIX statement

```
        KRONECKER (S1)(S2)(S3).
```

would be the following matrix on S3.

$$\begin{array}{cccc} 1 & 3 & 2 & 6 \\ 2 & 4 & 4 & 8 \\ -1 & -3 & 1 & 2 \\ -2 & -4 & 3 & 4 \\ 0 & 0 & .5 & 1.5 \\ 0 & 0 & 1 & 2 \end{array}$$

RECIPROCAL OF SQUARE ROOT OF DIAGONAL (mnemonic: RSD)

The RSD operation has two operands, an input address and an output address. The reciprocal of the square root of the main diagonal elements from the first matrix are used to form a single row vector which is output to the second operand address. Example:

```
            RSD (SEQ1)(SEQ3).
```

These programs were designed after the bulk of the MATRIX program description was typed.

II. Special Comments

A. Incore Address Option

Besides the standard SOUPAC addresses, MATRIX also recognizes the additional address I. The I symbol as an address represents internal storage in the machine.

An obvious use of this feature is to cut down on I/O time for matrices which are to be used in future operations within the current matrix program. The internal storage feature also saves time when the user desires his output from an operation to be printed or punched. The user must keep in mind that data cannot be passed to subsequent programs with the I storage. The user should also be aware of the restrictions on I storage as mentioned above in some of the subparameter operations (see INVERT, MULTIPLY, SQUARE, and TRANSPOSE). In all cases the use of this option is not recommended for matrices which do not fit within the memory available to the MATRIX program while running within any particular region size.

1) To add the matrix on SEQ1 to the matrix on SEQ2 leaving the result in core and also printing the result, code as follows:

```
ADD (SEQ1)(SEQ2)(I/PRINT).
```

2) To vertically augment the matrices in core, on SEQ1 and on SEQ2, storing the result on SEQ4, code as follows:

```
VERTICAL AUGMENT (I)(SEQ1)(SEQ2)(SEQ4).
```

B. Labeled Output

Provided in the MATRIX program is the facility to title and put column labels on any matrix which is printed using normal SOUPAC print conventions. The labeling feature is not allowed with the PRINT matrix operation.

To use the labeling feature, it is first necessary to put the title and labels in a temporary storage area. This is accomplished with the LABEL operation (see Subparameters).

To use a label which has been placed in a temporary storage area, code (L) after the print portion of the output address. If F format is also desired code either (F,L) or (L,F) after the print. Examples:

1) To move (copy) the matrix on SEQ1 onto SEQ5 printing the result in F format with title and column labels, code as follows:

```
MOVE (SEQ1)(SEQ5/PRINT(F,L)).
```

2) To add the matrices on SEQ1 and SEQ2 storing the result on SEQ3 and also printing the result with title and column labels, code as follows:

```
ADD (SEQ1)(SEQ2)(SEQ3/PRINT(L)).
```

3) To transpose the matrix stored on SEQ1 to SEQ2, printing out the result in F format with title and column labels, and punching out a card deck of the transposed matrix, code as follows:

```
TRANSPOSE (SEQ1)(SEQ2/PRINT(L,F)/X).
```

TRANSFORMATIONS

I. General Description

TRANSFORMATIONS is a data manipulation program. Unlike MATRIX, which performs operations on a complete matrix, TRANSFORMATIONS operates upon matrices one row at a time. This strategy provides for almost unlimited flexibility in transforming your data. Some of the general uses include creating new variables as functions of present variables, recoding or collapsing data, and reordering or eliminating variables. More advanced uses are facilitated by an instruction set which allows testing and branching depending on single variable characteristics or relations between variables, indirect addressing (FLAG-NOTATION), and inputting and outputting to and from different sequential units during the program.

TRANSFORMATIONS serves several purposes in the SOUPAC system. First, it can be used as a stand alone program to perform computations on your input data and yield the final results. Also, it can be used to prepare your data for input into another SOUPAC program or to make modifications from the output of one program for input into another.

II. Parameters

Parameter Number	Use or Meaning
1	Input address of the main input matrix. CARDS or SEQUENTIAL 1-15. This parameter can be left out if there is no input or if input comes from the input instruction.

The format of the entire TRANSFORMATIONS program follows:

```
TRANSFORMATIONS (C or Sn, n≤15).
:  subparameter cards which describe the transformation to be performed
•
END PROGRAM
```

III. Programming Considerations

The TRANSFORMATIONS program, as described in the first section, reads in one row of data at a time and then it executed the program until the end or until a last card instruction, and then it cycles on the identical program for each row of data.

There are 2000 variables allowed in the TRANSFORMATIONS program. Before each row of data is read into the program variables 1 through 1000 are filled with zeroes. Variables 1001 through 2000 are zeroed out only before the program begins so that during the program they can be used for accumulating sums or related totals as well as being used for temporary storage variables for row manipulations.

Input to the program can be specified by the parameter on the TRANSFORMATIONS card or by the INPUT instruction. Output can only be achieved by use of the OUTPUT instruction.

IV. Semi-table of Contents

V. Subparameter List

mnemonic	notes	operation name	examples
ABO		Abort	ABORT.
ABS	1	Absolute Value	ABS (1)(21).
ADD	1,2	Add	ADD (1)(2)(21). ADD (1)(3)(5)(7)(22).
ANG	1	Angle to Radians	ANG (1)(21).
A-C	1	Arccosine	A-C (1)(21).
A-S	1	Arcsine	A-S (1)(21).
A-T	1	Arctangent	A-T (1)(21).
C-G		Computed Go To	C-G (100)"A""B""C""D".
CON	3	Constant	CON (101)*4.3*. CON (102)(7).
COS	1	Cosine	COS (1)(21).
DIF	2	Difference if	DIF (1)(5)"A""B""C".
DIV	1,2	Divide	DIV (1)(2)(21). DIV (1)(2)(21)(3). DIV (1)(2)(21)(3)"A".
EBC	1	EBCDIC	EBC (1)(21).
EXC	1	Exchange	EXC (1)(2).
EXI		Exit	EXIT.
EXP	1	Exponent Base e	EXP (1)(21).
FAC	1	Factorial	FAC (1)(21).
FIX	1	Fixed Point Conversion	FIX (1)(21).
FLO	1	Floating Point Conversion	FLO (1)(21).
GO		Go to	GOTO "A".
IF		Arithmetic If	IF (1)"A""B""C".
INP		Input from Unit	INPUT (S2)(200).
LAS		Last	LAST.
ELO	1	Log base e	ELOG (1)(21).
LOG	1	Log base 10	LOG (1)(21).
MAX	1,2	Maximum Value	MAX (1)(2)(21). MAX (1)(3)(5)(7)(9)(22).
MIN	1,2	Minimum Value	MIN (1)(2)(21). MIN (1)(3)(5)(7)(9)(22).
MOD	1,2	Modular Arithmetic	MOD (1)(5)(21).
MOV	1	Move	MOVE (1)(21).

mnemonic	notes	operation name	examples
MUL	1,2	Multiply	MUL (1)(2)(21). MUL (1)(3)(5)(22).
NO		No operation	NOOP.
OUT	3	Output to Unit	OUT (S3)(1,50).
PER	3	Permute	PER (100)(5)(1,4)(8,9)(7).
RAD	1	Radians to Angle	RAD (1)(21).
RAI	1,2	Raise to Power	RAI (1)(2)(21).
REC	1,2	Recode	REC (1)"GT"(2)(3)*11*. REC (7)"EQ"*3*(7)*0*(7)*1*.
SIG	1	Sign Transfer	SIGN (1)(22).
SIN	1	Sine	SINE (1)(21).
SKI	2	Skip record on unit	SKIP (S2)(100). SKIP (S3)*1*.
SQU	1	Square root	SQU (1)(21).
SUB	1,2	Subtract	SUB (1)(2)(21).

XAD	-	add	
XDI	-	divide	These fixed point instructions are identical in operations performed and format with the corresponding floating point instructions, except that all variables appearing in these instructions must be in fixed point representation.
XIF	-	if	
XMU	-	multiply	
XSM	-	sum	
XSU	-	subtract	

1. For all parameters containing variables or floating point constants, DO notation may be used to indicate repetition of the operation on different sets of variables and/or floating point constants. (Explanation of DO-notation appears in Sec. 9).

2. All input parameters of the instruction which contain variables may be replaced by floating point constants.

3. For all parameters containing variables, DO-notation ranges may be substituted for a single variable occurance. (DO-notation ranges are described in SEC. 9)

VI. Transformations Labels

In TRANSFORMATIONS there are several instructions which perform some kind of test on your data. In most cases, the results of that test cause the program not to execute the next sequential instruction, but to branch to some other statement in the TRANSFORMATIONS program and continue executing with that statement. In order to refer to these statements we wish to transfer to, TRANSFORMATION'S labels are employed.

The form of these labels can be illustrated by the following example which skips over the divide statement if the divisor is zero.

```
IF(3)"NEXT" "JUMP" "NEXT".
"NEXT"DIV (2)(3)(4).
"JUMP"  next statement
```

<u>or</u>

```
IF(3) "*+1" "*+2" "*+1*.
DIV (2)(3)(4).
    next statement
```

The preceding equivalent examples exhibit the two types of TRANSFORMATIONS labels. The syntactical rules which govern the two types of labels follow.

Type 1

A. A type 1 label consists of eight or less alphanumeric characters set off by a pair of quotes.

B. Alphanumeric characters consist of the alphabet from A to Z and the numberic digits from 0 to 9.

C. Any unique label may appear as an operand or branch address of any number of TRANSFORMATIONS subparameter instructions.

D. Any label which appears as an operand or branch address of an instruction must appear immediately preceding and be part of at least one and only one TRANSFORMATIONS subparameter instruction in the present TRANSFORMATIONS program.

Type 2

A. A type 2 label consists of a positional reference of the form *+n set off by a pair of quotes.

B. The symbol * is pointing to the statement in which it appears. Therefore *+1 would point to the next statement, *+2 would skip one statement, and *-1 would point to the preceding statement.

C. This type of label need only appear as an operand or branch address and not before the TRANSFORMATIONS subparameter instruction which it is referencing.

D. Since the statement to which you are going to branch is always indicated relative to the statement from which you are branching, it is possible to point to an address which would lie beyond the end of the program or before the beginning of the program. Needless to say, this would result in an error condition.

E. Branching to "*+0" would create an infinite loop and is also illegal.

VII. Subparameter Description

ABO

The ABORT instruction causes immediate termination of the TRANSFORMATIONS program and the entire SOUPAC program. This instruction is often transferred to when internal tests reveal incorrect data. Example:

```
ABORT.
```

ABS

The ABSOLUTE VALUE instruction takes the absolute value of the first variable and stores it into the second variable. Example:

```
ABS (3)(25).
```

ADD

The ADD instruction has from three to one hundred parameters pointing to variables. The first variable through the next to last variable are summed and the result is stored into the last variable. Examples:

```
ADD (6)(7)(23).
ADD (1)(3)(5)(7)(9)(100).
```

ANG

The ANGLE TO RADIANS instruction converts the first variable, which should be a measure of an angle, into radians and stores the result into the second variable. Example:

```
ANG (3)(17).
```

A-C

The ARCCOSINE instruction takes the arccosine of the first variable and stores it into the second variable. The first vairable must be between minus one and one inclusive. The result will be stored in radians. Example:

```
A-C (7)(34).
```

A-S

The ARCSINE instruction takes the arcsine of the first variable and stores it into the second variable. The first variable must be between minus one and one inclusive. The result will be stored in radians. Example:

```
A-S (9)(13).
```

A-T

The ARCTANGENT instruction takes the arctangent of the first variable and stores it into the second variable. Example:

```
A-T (8)(70).
```

C-G

The COMPUTED GO TO has from two to twenty two parameters. The first parameter contains a variable and the following parameters contain labels. The basic form is

$$\text{C-G(v)"}L_1\text{" "}L_2\text{" "}L_3\text{" ...... "}L_n\text{".} \quad \text{where } n \leq 21$$

The variable must be floating point. If it is not of integral value then it is truncated, (all digits to the right of the decimal point are dropped). The instruction will then branch to the label whose position in the list is equal to the integral value of the variable. Example:

```
C-G(7)"A" "B" "C" "D" "E".
```

If variable 7 is equal to 4.0 or 4.3 then the instruction will branch to label "D".

CON

The CONSTANT instruction contains from two to one hundred parameters. The instruction is used to assign constant values to variables. The first parameter indicates either a variable or range of variables. The subsequent parameters contain the fixed point constant(s) and/or the floating point constant(s) which are assigned to the variable(s). Examples:

```
CON(7)(99) assigns fixed point value 99 to variable 7.
CON(8)*4.3* assigns floating point value 4.3 to variable 8.
CON(7,8)(99)*4.3* is equivalent to the previous two together.
```

The more complicated structures which contain ranges and increments for the variables and constants, uses DO-notation which is explained in Sec. 9. An example of the CONSTANT instruction with that structure will be included in that section.

COS

The COSINE instruction takes the cosine of the first variable and stores it into the second. The first variable must be expressed in radians. Example:

```
COS (1)(17).
```

DIF

The DIFFERENCE IF instruction contains five parameters. The first two parameters point to variables and the next three contain labels. The second variable is subtracted from the first. If the difference is negative the instruction branches to the first label, if the difference is zero it branches to the second or middle label, and if the difference is positive it branches to the third or last label. Example:

```
DIF (3)(5) "A" "B" "C".
```

If variable 3 minus variable 5 is negative the instruction will branch to label "A".

DIV

The DIVIDE instruction contains from three to five parameters. In the case of three parameters, the first variable is divided by the second variable and the result is stored in the third variable. A division by zero will terminate the program. Example:

```
DIV (1)(2)(30).
```

In the case of four parameters, the division will take place as normal except when the second variable is zero. In that event, the fourth variable will be stored into the third variable as a supplied quotient. Example:

```
DIV (1)(2)(30)(31).
```

If the fifth parameter is added, it indicates a label to be branched to in the event of a division by zero. The branch will take place after the supplied quotient is stored into the third variable. Example:

```
DIV (1)(2)(30)(31)"ZERO".
```

EBC

The EBCDIC instruction converts characters, which are read into the program by means of an A1 format field, into floating point numbers. The first variable contains the character. The result after the table look up will be stored into the second variable. Example:

```
EBC (6)(89).
```

The table used for the conversions is located following the subparameter descriptions. This instruction is often used to prepare character codes for input to a SOUPAC FREQUENCY program.

EXC

The EXCHANGE instruction exchanges the contents of two variables. Example:

```
EXC (1)(2).
```

EXI

The EXIT instruction causes immediate termination of the TRANSFORMATIONS program, but will continue to execute the SOUPAC program which follows. Example:

EXP

The EXPONENT BASE e instruction raises e to the power of the first variable and stores the result in the second variable. Example:

```
EXP (31)(704).
```

FAC

The FACTORIAL instruction calculates the factorial of the first variable and the result is stored in the second variable. Example:

```
FAC(87)(120).
```

FIX

The FIXED POINT CONVERSION instruction converts the floating point variable indicated by the first parameter and stores the result into the second variable. Example:

```
FIX (2)(9).
```

FLO

The FLOATING POINT CONVERSION instruction converts the fixed point variable indicated by the first parameter and stores the result into the second variable. Example:

```
FLOAT (7)(8).
```

GO

The GO TO instruction unconditionally branches to the label indicated by the only parameter. Example:

```
GO TO "LABEL".
```

IF

The IF statement has four parameters. The first parameter contains a variable and the remaining three parameters are labels. If the variable is negative, zero, or positive the instruction will branch to the first, second, or third label respectively. Example:

```
IF (33) "L1""L2""L3".
```

If variable 33 is positive the instruction will branch to label "L3".

INP

By use of the INPUT instruction you can read in additional rows of data in addition to the rows from the main input matrix. The first parameter indicates which sequential unit the data will be input from. The second parameter indicates the starting variable number where you wish the row to be read in. Example:

```
INPUT (S3)(500).
```

If S3 contains 64 variables then this input instruction will read the next row from S3 and store it in variables 500 through 563. Be careful to avoid overwriting of existing variables which you need and also of trying to read more or less rows than exist on a particular unit.

LAS

The LAST CARD instruction allows instructions to be performed after the last row of data has been read in and processed. The LAST instruction divides a program into regular and last card segments. The regular section, as is a TRANSFORMATIONS program without the LAST option, is executed once for every row of data. After all the main input data is processed the last card segment is executed once. One of the main uses of the LAST instruction is to analyze data accumulated in variables 1001-2000 during the regular segment. Example:

```
LAST.
```

Only one LAST instruction may be used for TRANSFORMATIONS programs and branching between regular and last card segments is prohibited.

ELO

The LOG BASE e instruction takes the natural log of the first variable and stores it in the second variable. Example:

```
ELOG (3)(17).
```

LOG

The LOG BASE 10 instruction takes the base 10 log of the first variable and stores it into the second variable. Example:

```
LOG (18)(34).
```

MAX

The MAXIMUM VALUE instruction has from three to one hundred parameters pointing to variables. The variable with the largest value from the first variable to the next to last variable is stored into the last variable. Examples:

```
MAX (1)(2)(7).
MAX (1)(7)(8)(11)(15).
```

MIN

The MINIMUM VALUE instruction has from three to one hundred parameters pointing to variables. The variable with the smallest value from the first variable to the next to last variable is stored into the last variable. Examples:

```
MIN (11)(13)(29).
MIN (1)(3)(4)(7)(8)(10)(12).
```

MOD

The MODULAR ARITHMETIC instruction finds the value of the first variable modulus the second variable and stores the result into the third variable. Example:

```
MOD (1)(4)(7).
```

MOV

The MOVE instruction stores a copy of the first variable into the second variable. If a value already exists in the second variable it will be overwritten. Example:

```
MOVE (3)(9).
```

MUL

The MULTIPLY instruction has from three to one hundred parameters pointing to variables. The first variable through the next to last variable are multiplied together and the result is stored in the last variable. Examples:

```
MUL (1)(2)(20).
MUL (3)(4)(5)(6)(10).
```

NO

The NOOP instruction does nothing. Its primary use is when it is preceded by a label and used as a placeholder in the TRANSFORMATIONS subprogram to which many different instructions branch. It is commonly used at the end of a TRANSFORMATIONS subprogram where several isolated groups of instructions all wish to branch to the end. Example:

```
"END" NOOP.
```

OUT

The OUTPUT TO UNIT instruction is the only way the TRANSFORMATIONS program can output a row of data. There are two parameters. The first indicates the unit to which the row of variables should be output. The second parameter can be either a single variable or a range of variables. Examples:

```
OUT (S2)(7).
```

This will output onto S2 a row with one variable.

```
OUT (S4)(8,14).
```

This will output onto S4 a row with seven variables, variable eight through variable fourteen. All outputs to the same unit must have the same number of variables output. A more detailed description of types of ranges which are allowed will appear in Sec. 9 on DO notation.

PER

The PERMUTE instruction permutes the order of all or a subset of your variables. It can have from two to one hundred parameters, all indicating variable numbers. The first parameter indicates a starting point of where a string of variables should be placed. The rest of the parameters compose that string of variables. Each of the parameters in that string represent either a single variable or a range of variables. Examples:

```
PER (100)(3)(10,14)(4,5).
```

This example places variable 3 in variable 100, variables 10 through 14 in variables 101 through 105, and variables 4 and 5 in variables 106 and 107. A more detailed description of types of ranges which are allowed will appear in Sec. 9 on DO notation ranges.

```
PER (2)(1,1999).
```

This example will not propagate variable 1 through all the variables. It will perform the intended purpose of raising all variable numbers up one.

RAD

The RADIANS TO ANGLE instruction converts the first variable, which should be a measure of radians, into an angle and stores the result into the second variable. Example:

```
RAD (9)(13).
```

RAI

The RAISE instruction raises the first variable to the power contained in th second variable and stores the result into the third variable. Example:

```
RAISE (1)(2)(7).
```

REC

The RECODE instruction recodes variables depending on the satisfaction of a set of conditions. The sequence of parameters depends on the number of conditions that must be met. The RECODE instruction introduces a new set of terminology which follows:

A. RELATIONAL OPERATORS

"LT"	less than
"LE"	less than or equal to
"EQ"	equal
"NE"	not equal
"GE"	greater than or equal
"GT"	greater than

B. CONNECTIVES

"AND"
"OR"
"EOR"

C. CONDITION SET

A condition set consists of three parameters, the first and third parameters pointing to variables and the second parameter containing a relational operator. Any condition set is either true or false depending upon whether the two variables satisfy the conditions of the relational operator.

D. RECODE SET

A recode set consists of two parameters. The first parameter indicates a variable. The second parameter indicates a variable or a floating point constant. If a recode set is executed the value of the second variable or floating point constant is stored into the first variable.

The RECODE instruction consists of from one to twenty one condition sets joined together in the case of more than one by connectives. This is followed by a recode set to be executed if the logical product of the condition sets is true and optionally a second recode set to be executed if the logical product is false. Examples:

```
REC (4) "EQ" *3* (4) *1* (4) *0*.
```

If variable 4 equals 3.0 then recode it to 1.0, if not then recode it to 0.0.

```
REC (6) "GE" (10) "AND" (7) "LE" (20)(110) (111)
```

If variable 6 is greater than or equal to variable 10 and variable 7 is less than or equal to variable 20 then recode variable 110 to the value of variable 111.

SIG

The SIGN instruction places the sign of the first variable on the second variable. It is often used for saving signs of variables during intermediate calculations. Example:

```
SIGN (1)(901).
```

SIN

The SINE instruction takes the sine of the first variable and stores it into the second variable. Example:

```
SIN (1)(11).
```

SKI

The SKIP instruction has two parameters. The first parameter is a sequential unit. The second is a variable or floating point number. The number in the variable or the floating point number indicate the number of rows to be skipped on the specified unit. Example:

```
SKIP (S2)(7)
SKIP (S5) *1*.
```

SQU

The SQUARE ROOT instruction takes the square root of the first variable and stores it into the second variable. Example:

```
SQU (13)(14).
```

SUB

The SUBTRACT instruction subtracts the second variable from the first variable and stores the result into the third variable. Example:

```
SUB (7)(9)(10).
```

XAD - fixed point addition

XDI - fixed point divides

XIF - fixed point if

XMU - fixed point multiply

XSM - fixed point sum

XSU - fixed point subtract

The preceding fixed point instructions have the same parameters as the corresponding floating point instructions. The variables used in the fixed point instructions must be the fixed point representation rather than the normal floating point representation.

VIII. EBCDIC Conversion Table

Character	Floating Point Number	Character	Floating Point Number
blank	-0.	¢	74.
0	0.	.	75.
1	1.	<	76.
2	2.	(	77.
3	3.	+	78.
4	4.	|	79.
5	5.	&	80.
6	6.	!	90.
7	7.	$	91.
8	8.	*	92.
9	9.	)	93.
A	10.	;	94.
B	11.	¬	95.
C	12.	-	96.
D	13.	/	97.
E	14.	,	107.
F	15.	%	108.
G	16.	_	109
H	17.	>	110.
I	18.	?	111.
J	19.	:	122.
K	20.	#	123.
L	21.	@	124.
M	22.	'	125.
N	23.	=	126.
O	24.	"	127.
P	25.		
Q	26.		
R	27.		
S	28.		
T	29.		
U	30.		
V	31.		
W	32.		
X	33.		
Y	34.		
Z	35.		

IX. DO-Notation

DO-notation is a facility provided in the TRANSFORMATION program to enable a user to easily and compactly perform an operation on a set of variables instead of performing that operation on each of the variables individually. This concept of DO-notation corresponds to the concept of FORTRAN DO-loops. The following form of DO-notation would be used in a parameter which points to a variable.

(V_1, V_2, I)

V_1 = the initial variable of the set

V_2 = the criterion variable for termination of the set.

I = the increment

Examples:

(1, 5, 1) points to variables 1, 2, 3, 4, and 5.

(1, 5, 2) points to variables 1, 3, and 5.

(4, 14,3) points to variables 4, 7, 10, and 13.

If the increment is not specified it is assumed to be 1, Example:

(2, 10, 1) is equivalent to (2, 10).

A. The major use of DO-notation is to indicate repetition of an instruction on different sets of variables. Examples:

```
 ADD(1,5,2)(6,8)(12,14)  is equivalent to  ADD(1)(6)(12).
                                           ADD(3)(7)(13).
                                           ADD(5)(8)(14).

MUL(1,4)(100)(101,107,2)  is equivalent to  MUL(1)(100)(101).
                                            MUL(2)(100)(103).
                                            MUL(3)(100)(105).
                                            MUL(4)(100)(107).
```

If, in an instruction containing several parameters using DO notation, the sets of variables are unequal in length, the instruction will cycle until the longest set of variables has been satisfied. The variables used in the shorter sets after they have been exhausted will be the last variables of that set. Example:

```
MAX(1,7,3)(14,15)(13,15,2).  is equivalent to  MAX(1)(14)(13).
                                               MAX(4)(15)(15).
                                               MAX(7)(15)(15).
```

B. The secondary use of DO-notation, called DO-notation ranges, is used with the constant, output, and permute instructions. Instead of indicating a repetition of the instruction on each variable in the set, the instruction is executed once. The parameter in which the

DO-notation range occurs now points to a string of variables. Examples:

OUTPUT(S1)(2,7). outputs the string of variables 2,3,4,5,6 and 7.

OUTPUT(S2)(3,7,2). outputs the string of variables 3,5, and 7.

PERMUTE (100)(3)(10,15)(20,24,2).
has two parameters of DO-notation ranges in one instruction. The instruction places into variables 100 through 109, the following variables: 3, 10,11,12,13,14,15, 20,22,24

Looking at the CONSTANT instruction, we see DO-notation ranges used to indicate strings of fixed and floating point constants as well as strings of variables.

The first parameter of CONSTANT can point to a variable or a string of variables. The later is the only case which involves DO-notation ranges, so the discussion will be confined to that case.

CON(2,4)(1)(2)(3). places the fixed point constants one, two and three, into the string of variables 2,3,4

By using DO-notation ranges with fixed point constants the equivalent instruction would be

CON(2,4)(1,3).

The next two examples are also equivalent. They both assign to variables five through eight the floating point constants 2.5,5.0,7.5, and 10.0.

CON(5,8)*2.5**5.0**7.5**10.0*.
CON(5,8)*2.5,10.0,2.5*.
The initial values of the string of floating point constants is 2.5. The termination criterion is 10.0. The increment is 2.5

All three types of DO-notation ranges can be used together in the CONSTANT instruction. The following example combines both of the preceding sets of examples into one instruction.

CON(2,8)(1,3)*2.5,10.0,2.5*.

Note: If, when using DO-notation ranges with the CONSTANT instruction, the string of variables is unequal in length to the total number of fixed and/or floating point constants indicated, the string of constants is truncated if it is longer and if it is shorter the last constant is assigned to the remainder of the variables.

X. Flag Notation

Flag notation is TRANSFORMATION's version of indirect addressing. Instead of a parameter pointing directly to a variable, it points indirectly to a variable through another variable. The parameter points to a variable which in turn points to another variable. This feature enables an instruction to point to different sets of variables depending upon the values assigned to the intermediate variables.

The main type of flag notation is called F-flag notation. F-flag notation is indicated by inserting an F directly after the variable number. Example:

```
ADD(7F)(8F)(9F).
```

Restriction: The values in the intermediate variables of flag-notation (variables 7,8, and 9 in this example) must be in fixed point representation and must point to a valid variable number.

Let the notation Vn indicate variable n. Example: V7 indicates variable 7.

```
If     V7 = 100
       V8 = 150
       V9 = 180
```

then the preceding example would generate after the indirect addressing takes place:

```
ADD(100)(150)(180).
```

F-flag notation can also be used with DO-notation. Example:

```
If     V100 = 2    V110 = 6    V120 = 10
       V101 = 13   V111 = 16   V122 = 19
       V102 = 5    V112 = 9    V124 = 13
```

then MUL(100F,102F)(110F,112F)(120F,124F,2). would generate after indirect addressing.

```
MUL(2)(6)(10).
MUL(13)(16)(19).
MUL(5)(9)(13).
```

Note that the limits of the DO-notation are extracted form the intermediate variables and not from the final variables.

The other type of flag notation is called D-Flag notation. Its only use is with DO-notation. The difference between it and F-flag notation is that the limits of the DO-notation with D-flag notation are derived from the final variables after indirect addressing takes place. Example:

Using the same variable values as above.

```
MUL(100D,102D)(110D,112D)(120D,124D).
```

would generate after indirect addressing:

```
MUL(2,5)(6,9)(10,13).
```

or

```
MUL(2)(6)(10).
MUL(3)(7)(11).
MUL(4)(8)(12).
MUL(5)(9)(13).
```

XI. A TRANSFORMATIONS Example

Let us consider a set of data consisting of ten variables. Variable one contains a zero for females and a one for males. Variables two through five and eight through ten all range from zero to ninety-nine and are to be collapsed into the numbers one to four representing quartiles. Variables six and seven are to be recoded into a dichotomous variable depending if the value is five or not five. The output is then split into males and females suitable for input into two separate FREQUENCY programs. Also desired are averages from variables two through ten before recoding takes place. Totals are kept during the regular segment and then in the last card segment these are divided by the sample size and printed.

```
TRANSFORMATIONS
INPUT (C)(1).                           inputs one row from cards
PERMUTE (6)(8,10)(6,7).                 reorders the variables
ADD (1001)*1*(1001).                    increments row number by one
ADD (1002,1010)(2,10)(1002,1010).       adds respective values to row totals
RECODE (2,8)"LT"*25*(2,8)*1*.           recodes first quartile values
RECODE (2,8)"GE"*75*(2,8)*4*.           recodes fourth quartile values
RECODE (2,8)"GE"*50*(2,8)*3*(2,8)*2*.   recodes third and second quartile values
RECODE (9,10)"EQ"*5*(6,7)*1*(6,7)*0*.   creates dichotomous variables
IF (1)"BAD""FEMALE""MALE".              branches based on male or female
"BAD"ABORT.                             aborts program due to bad data
"FEMALE"OUTPUT(S2)(2,10).               outputs female data
GO TO "END".                            branches around male output
"MALE"OUTPUT(S3)(2,10).                 outputs male data
"END"NOOP.                              this instruction is only a placeholder
LAST.                                   indicates beginning of last card segment
DIVIDE (1002,1010)(1001)(1002,1010).    calculates averages
OUTPUT (P)(1001,1010).                  prints sample size and averages
END PROGRAM
```

The regular segment is executed once for every row of card input data. Here the values are recoded and output, while also row totals are kept in variables greater than 1000. The last card segment is executed once to calculate and print the averages.

XII. NOTES and IDEAS

1. Missing data of the form -0.0 can be differentiated from 0.0 only in the recode statement, so if this distinction must be made, -0.0 should first be recoded to another value before testing.

2. The valid outputs in the OUTPUT instruction are PRINT and/or Sn, $n \leq 15$.

3. When collapsing data be careful not to overlap your recoding and inadvertently recode values twice or more.

4. The former ZAP and SPRAY instructions have been incorporated into the CONSTANT instruction by using DO notation with floating point constants.

5. The former RECIPROCAL instruction can now be performed using a floating point constant of *1* in the DIVIDE instruction.

6. The ONE RECODE, TWO RECODE, OPEN WHEN, and CLOSED WHEN instructions have all been replaced by the RECODE instruction. For those not familiar with the new terminology in the RECODE instruction more examples appear below.

```
Given:     V1 = 10    V3 = 13    V5 = 23
           V2 = 23    V4 = 89    V6 = -7
```

The following condition sets have the respective truth values.

```
(1) "GT" (2) is false
(3) "LE" (4) is true
(2) "EQ" (5) is true
(6) "GE" *0* is false
(2) "NE" (5) is false
(6) "LT" (4) is true
```

If two or more condition sets are joined by "AND", they must all be true for the logical product to be true. If two or more condition sets are joined by "OR", then the logical product is true if any of the condition sets are true. If the connectives are mixed, then the "AND" connective is of higher precedence than the other connectives in the same way that multiply is of higher precedence than addition in ordinary arithmetic.

BASIC POPULATION STATISTICS PACKAGE

FREQUENCY COUNTING AND MEASURES ƆF ASSOCIATION

I. General Description

This program computes tables of the frequency of occurrence of values that input variables take, and where appropriate, measures of association may be computed. Input to the program may be in the form of previously computed tables (on which measures of association will be computed) or may be in the form of raw data. Only integer numbers may be counted; decimal point data will be rounded. Negative values are allowed.

A. FREQUENCY COUNTING

The following options are available:

1. Either one-dimensional or two-dimensional tables may be specified. For one-dimensional counts, the frequency of occurrence for each value of the variable is listed. For two-dimensional counts, each value of the second variable is counted separately for each value of the first variable.

2. Control variables may be used which enable counting to be done in up to 12 dimensions. If control variables are specified, data must be presorted on these variables. When the value of any variable designated a control variable changes from one row to the next, counting is stopped and a new table is started. Thus counting proceeds as long as the values of all control variables remain constant.

3. The minimum and maximum values to be attained may be specified separately for each variable to be counted (or, optionally, not specified at all). If either the maximum or minimum values are not specified, they will be determined from the data using an extra read of the data. Values which fall below the minimum or above the maximum are ignored. This capability adds flexibility to the program and may be an appreciable cost saver. Its misuse by gross estimates of minimum and maximum values can, however, be costly.

4. For each cell in a one-dimensional table, the percentage, if requested, of the total sample that were counted there will be printed. For two dimensional tables, the percentage of the row and column may also be requested.

5. A weighting variable may be specified. Without a weighting variable, frequency counts are advanced by one for each occurrence of a value. When a weighting variable is used, the frequency counts are advanced by the value of the weighting variable for the row. Thus some rows of data may be given more importance than others.

6. Labels can be given for variables so that output is more readable. Each label is restricted to eight characters or less.

7. Input may be from previously computed two-dimensional tables, from which measures of association can be directly computed.

B. MEASURES OF ASSOCIATION

The following coefficients are calculated and printed on option for two-way tables:

1. Chi-square and related coefficients

Let: n = total population of the table

n_{ab} = number of Vertical classification a (column a) and Horizontal classification b (row b)

$$n_{a.} = \sum_{b} n_{ab}$$

$$n_{.b} = \sum_{a} n_{ab}$$

α = number of rows
β = number of columns

Then: $$\text{chi-square} = \sum_{ab}\sum \frac{\left(n_{ab} - \frac{n_{a.}\, n_{.b}}{n}\right)^2}{n_{a.}\, n_{.b}/n}$$

adjusted chi-square (Yate's correction for continuity) for 2 x 2 tables only =

$$\sum_{ab}\sum \frac{\left(n_{ab} - n_{a}\, n_{b} - 1/2\right)^2}{\frac{n_{a}\, n_{b}}{n}}$$

$$C = \left(\frac{\text{chi-square}/n}{1 + \text{chi-square}/n}\right)^{1/2}$$

$$T = \left(\frac{\text{chi-square}/n}{(\alpha-1)(\beta-1)}\right)^{1/2}$$

C and T are measures of contingency and can be looked up in contingency tables. The maximum expected frequency is also printed.

2. Lambda coefficients

Let: $n_{am} = \underset{b}{\text{Max}}\ n_{ab}$

$n_{mb} = \underset{a}{\text{Max}}\ n_{ab}$

$n_{.m} = \underset{b}{\text{Max}}\ n_{.b}$

$n_{m.} = \underset{a}{\text{Max}}\ n_{a.}$

$$\text{Lambda} = \frac{\sum_a n_{am} + \sum_b n_{mb} - n_{.m} - n_{m.}}{2n - n_{.m} - n_{m.}}$$

$$\text{Lambda H} = \frac{\sum_a n_{am} - n_{.m}}{n - n_{m.}}$$

$$\text{Lambda V} = \frac{\sum_b n_{mb} - n_{m.}}{n - n_{m.}}$$

Lambda coefficients will be indeterminate if all values lie in one column or row.

Lambda H can be defined as the decrease in probability of error in predicting the H-variable when knowledge of the value of the V-variable is considered as opposed to random guessing of the H-variable.

Ninety-five per cent confidence limits are calculated and printed for Lambda H and Lambda V using the methods discussed by Goodman and Kruskal in their second article. (See references). Lambda is always between Lambda H and Lambda V.

3. Weighted Lambda Coefficients

$$\text{Weighted Lambda H} = \frac{\sum_a \frac{n_{am}}{n_{a.}} - \underset{b}{\text{Max}} \sum_b \frac{n_{ab}}{n_{a.}}}{\alpha - \underset{b}{\text{Max}} \sum_a \frac{n_{ab}}{n_{a.}}}$$

$$\text{Weighted Lambda V} = \frac{\sum_b \frac{n_{mb}}{n_{.b}} - \underset{a}{\text{Max}} \sum_b \frac{n_{ab}}{n_{.b}}}{\beta - \underset{a}{\text{Max}} \sum_b \frac{n_{ab}}{n_{.b}}}$$

These are Lambda H and Lambda V calculated using weighted quantities $1/\alpha \frac{n_{ab}}{n_{a.}}$ and $1/\beta \frac{n_{ab}}{n_{.b}}$, respectively, instead of n_{ab}.

No confidence limits are provided for the weighted lambda coefficients.

4. Gamma Coefficient

Let: $$PS = \sum_{ab}\sum n_{ab} \left[\sum_{a'>a} \sum_{b'\atop b>b} n_{a'b'}\right]$$

$$PD = \sum_{ab}\sum n_{ab} \left[\sum_{a'>a} \sum_{b'>b} n_{a'b'}\right]$$

Then $$\text{Gamma} = \frac{PS - PD}{PS + PD}$$

Ninety-five per cent confidence limits for Gamma are calculated using the method outlined and preferred in the second article by Goodman and Kruskal.

C. References

These coefficients are discussed and compared by Leo A. Goodman and William H. Kruskal in their article "Measures of Association for Cross Classification", American Statistical Association Journal, December, 1954.

The Gamma coefficient is their suggested measure.

The C coefficient was first suggested by Karl Pearson and the T coefficient is due to Tchuprow.

The Lambda coefficients apparently were first suggested by Louis Guttman ("The Predication of Personal Adjustment", Bulletin 48, Social Science Research Council, New York, 1941).

The development of the approximate sampling theory and of the machinery for calculating the confidence intervals for Lambda and Gamma was done in a sequel article by Goodman and Kruskal: "Measures of Association for Cross Classification III; Approximate Sampling Theory", American Statistical Association Journal, June, 1963.

The statistics that are requested will be printed immediately following each table.

II. Restrictions

The program is limited currently to 450 input variables and 1000 tables. Tables are restricted to a maximum of 80,000 cells, each of which can hold a maximum count of 32,767. As many tables as will fit into work storage (80,000) will be computed in each read of the data. If tables will fit into 80,000 cells, card input is allowed. If maxima and minima are not specified for card input, data will be transferred to disk during preread of data.

III. Parameters

A. Main Parameter Card

Immediately following the program name FREQUENCY (mnemonic: FRE), the following parameters are listed, each enclosed in parentheses with a period after the last parameter used:

Parameter Number	Use or Meaning
1	Input Address.
2	0 - ignore blanks 1 - count blanks separately 2 - count blanks as zeroes
3	Spacing 0 - normal spacing 1 - one table per page
4	Address of labels.
5	Variable number of weight variable.
6	Type of input 0 - raw data n - where n is the number of previously computed tables. If n > 1, then input must be from cards, and each table is a separate data deck.

If both parameters 1 and 4 are cards, the labels must precede data.

B. Subparameters

Subparameters follow the main parameter card and can be in any order. A period must follow each subparameter statement though the statement can be continued on more than one card. If the subparameter statement is left out, the option is not used. In the following explanation I = integer and F = real number.

Mnemonic	Use or Meaning
PER(I)(I)(I).	Per cents are requested 0 - no 1 - yes 1st integer = total per cent 2nd integer = row per cent 3rd integer = column per cent
MIN*F**F*....... . MAX*F**F*....... .	Minimum and maximum are given. The last value is propagated to any remaining variables. Data will be reread if either MIN or MAX is missing.

Mnemonic	Use or Meaning
MEA(I)(I)(I)(I). Only applicable to 2-way tables	Measures are requested 0 - no 1 - yes 1st integer = X^2 (with a code of 2 both X^2 and a table of expected frequencies will be printed) 2nd integer = λ (lambda) 3rd integer = weighted λ 4th integer = γ (gamma)
CONTROL(I)(I)..... .	Up to 10 control variables are allowed. The I's should be the variable numbers of the control variables.
ONE(I,I,I)(I,I,I)..... . TWO(I,I,I)(I,I,I)..... .	One and only one of these two must be in every program. ONE means one-way tables. TWO means two-way tables. In ONE, (I,I,I) specifies one range of tables. In TWO, (I,I,I)(I,I,I) specifies one range of 2-dimensional tables.

The notation (I,I,I) has the following meaning: If it is absent completely, i.e., ONE. or TWO. then all possible tables are calculated. The first integer is the initial value, the second is the terminal value and the third is the increment. It means: take all values starting at the first integer and stepping by the third integer until you reach the second integer. If the third integer is missing, the increment is taken to be one. If the second is also missing, then the first is taken as a single table specification. As many as wanted can be specified subject to the following restrictions: In the two-way tables, no more than 540 separate ranges, i.e., (I,I,I)(I,I,I) can be specified.

IV. Labels

Labels can come from cards or temporary storage. Each label should be treated as if it were two variables each 4 characters long. For example, if there are 6 variables in the input data, then there would be twelve variables for labels and the data card would be DATA(12)(12A4).

All the labels are treated as one row of input n variables long. Labels need not be given for each variable but if a variable is skipped and more labels follow, then it should be replaced with eight blanks.

V. Examples

1.
```
FRE(C).
PER(1).
CONTROL(1)(3).
ONE(2,6,2)(9).
END P
```

Input is from cards; per cent of totals will be printed; control variables are 1 and 3; resulting tables are 2, 4, 6, and 9. Blanks will be ignored.

```
2.  FRE(C)(2)(1)(C)(2).
    PER(1)(1)(1).
    TWO(3)(4)(2,6,2)(1,5,2).
    END P
```

All per cents will be given; 2 will be the weighting variable; resulting tables will be 3 vs 4; 2 vs 1; 2 vs 3; 2 vs 5; 4 vs 1; 4 vs 3; 4 vs 5; 6 vs 1; 6 vs 3; 6 vs 5. Tables will be printed one per page and blanks will be counted as zeroes.

Since both labels and data are on cards the deck will look like this:

```
FRE(C)............   .
:
:
END S
DATA(n)(nA4)
label for first labeled variable..........label for last labeled variable
END#
DATA(n/2)(.....)
:
:
END#
```

```
3.  FRE(S1)(1).
    TWO.
    MEA(2)(1)(1)(1).
    END P
```

All possible two-way tables will be calculated; all four measures will be calculated and the table of expected frequencies will be printed. Blanks will be counted separately.

```
4.  FRE(C)()()()()(2).
    MEA(1)()(1).
    END P
```

Input is in the form of two previously computed tables. X^2 and weighted λ will be calculated.

Since there are two tables the deck will look like this:

```
FRE(C)............   .
:
:
END S
DATA...............
:
:
END#
DATA...............
:
:
END#
```

VI. Output Examples

A. ONE-DIMENSIONAL TABLE

A one-dimensional frequency table might be output as follows:

```
VALUE     1     4     5     9     TOTAL
FREQ      3    25    30     2       60
```

This table indicates that the value 1 occurred 3 times, that the value 4 occurred 25 times, and so on, for a total of 60.

B. TWO-DIMENSIONAL TABLE

A two-dimensional frequency table might look like this:

```
VARIABLE 1 ACROSS
VARIABLE 2 DOWN

   VALUE    1     3     5
     2      1     4     3
     5      2     8     1
     7      1     2     1

                          SAMPLE SIZE = 23
```

This table would indicate that simultaneous observations of 1 for variable 1 and 2 for variable 2 occurred once. A value of 3 for variable 1 at the same time as a value of 5 for variable 2 occurs 8 times. The number of observations in the sample was 23.

RANK ORIERING PROGRAM

I. General Description

A. Purpose

The RANK ORDERING program receives as input raw data matrix and produces as output a matrix in which each element has been replaced by a number denoting the rank of the element WITHIN ITS COLUMN. In other words, each column of the input matrix is considered a separate variate and will be converted to a corresponding ranking.

The smallest variate-value is assigned rank 1.0, the next largest a rank 2.0, etc., until the largest variate-value is assigned the highest rank. In the case of tied values, identical ranks are assigned to equal values, the rank-number being set equal to the average of the rank which would occur if the tied values were distinguishable. This is sometimes known as "mid-rank method".

B. References

Kendall, Maurice G., Rank Correlation Methods, Charles Griffin and Co., Ltd. London, 1948.

II. Restrictions

A. Input

The input data to this program may come from any source. If cards are used as input, the number of rows in the input matrix must be specified on the data format card and the total number of elements in the matrix may not exceed 30,000. The maximum number of rows for any matrix input to this program is 30,000, and the maximum number of columns for any matrix input to this program is 450.

B. Output

If an input matrix contains more than 30,000 elements, an automatic partitioning of the input data occurs such that each partition contains the maximum number of complete columns possible within the constraint that no one partition may contain more than 30,000 elements.

The results of the ranking of each partition are output separately, one partition per output address specified as a parameter on the program parameter card. A maximum of twenty-one such output address are allowed.

CAUTION: If partitioning is anticipated, the user should specify one output address for each partition anticipated. This warning applies especially in the case where printed or punched output occurs. Printing and punching will occur only for the partitions for which it is specified.

The exception is for partitions over the twenty-first one. For partitions beyond the twenty-first, printing and punching is done if it was specified for the twenty-first partition. However, no partitions beyond the twenty-first one may be stored on a peripheral device (SEQUENTIAL address).

C. Data

Since all comparisons in this program are done in single word length operands, in some cases the program may not be able to successfully differentiate between two values which agree through the first five significant digits and differ in subsequent digits.

III. Parameters

The parameters for the RANK ORDERING program must follow the program name on the program call card in the order given below:

Parameter Number	Use or Meaning
1	Input Address.
2-23	Output Address.

IV. Special Comments

If RANK ORDER correlation coefficient ρ (Spearman's rho) is desired, the rankings should be input to the CORRELATION program (see individual program description) and the Product Moment Correlation coefficient obtained.

CAUTION: Input to the CORRELATION program is restricted to 175 columns (variables).

STANDARD SCORES

I. General Description

This program is used to calculate the following:

Mean: $\bar{X}_j = \frac{\sum_{i=1}^{N} X_{ij}}{N}$

Standard Deviation: $S_j = \left[\frac{N \sum_{i=1}^{N} X_{ij}^2 - (\sum_{i=1}^{N} X_{ij})^2}{N^2}\right]^{1/2}$

Variance: $V_j = S_j^2$

Standardized Scores: $Z_{ij} = \frac{X_{ij} - \bar{X}_j}{S_j}$

Moving Averages: $\bar{X}_j = \frac{\sum_{i=1}^{b} X_{ij}}{b}$ where b=number of periods

II. Restrictions

A. The maximum number of variables is 450.

B. Means, standard deviations, and N's may be calculated using as many as 30 control variables. Data must be presorted (for instance with SORT-MERGE or on a card sorting machine) on the control variables.

C. In obtaining moving averages where nvar = number of variables and b = length of period, nvar*b is fixed for any core size. If a design will not fit, a message will be printed giving proper increment for Region. At 200K approximately 12000 cells (nvar*b) are available.

D. Moving averages are exclusive of all other options.

E. Output is of four categories:

1. With or Without Control Breaks

   a. Sample size, mean, standard deviation and variance.

   b. Moving averages: sample size, mean, standard deviation and variance.

2. Without Control Breaks

   a. Standard scores, printed output includes sample size, mean, standard deviation and variance. Missing data option for mean, etc., is possible, but user is cautioned against its use. Means, etc., may be output to a temporary storage location.

b. Standard scores about a given mean and standard deviation. Printed output includes sample size, mean, standard deviation and variance. Missing data option for mean, etc., is possible. Output may consist of standard scores and about a given mean and standard deviation. Output may be print and/or two different temporary storage locations. Means, etc., may be output to a temporary storage location.

III. Parameters

The parameters appear on the program call card following the program name STANDARD SCORES in this order:

Parameter Number	Use or Meaning
1	Input Address. SEQUENTIAL 1-15. Cards if only means, standard deviations, and variance desired, or if precalculated means and standard deviation are supplied (see parameter 10).
2	Output Address of Standardized Scores.
3	Output Address for Mean, Sample Size, Standard Deviation, and Variance. Sample Size, Mean, Standard Deviation, and Variance can be put out on a temporary unit. Output is in the form of four column vectors (N, $\overline{X}$, S_j, V_j).
4	If 1, use N-1, if 0. use N for denominator of standard deviations. N-1 gives an unbiased estimate of the population standard deviation and population variance. N gives the sample standard deviation and sample variance.
5	Output Address for Standard Scores about a specified Mean and Standard Deviation. SEQUENTIAL 1-13 and/or PRINT.
6	If parameter 5 is being used, place desired Mean between asterisks, for example, *50*.
7	If parameter 5 is being used, place desired Standard Deviation between asterisks, for example, *5*.
8	Moving Averages: Put the number of periods (observations) over which it is desired that the data be averaged (i.e. b). If control variables are being used and/or the actual number of observations is less than stated, the data will be averaged using the actual number of observations.

Parameter Number	Use or Meaning
9	If set equal to 1, blanks coded as -0.0 will be checked for. $\bar{X}_j$, S_j, and V_j will reflect reduced N.
10	Input Address of means and standard deviations. 1st row contains means, 2nd row contains standard deviations. All additional rows input are ignored. Valid only for standard scores.

If using controls, on a separate card immediately after the STANDARD SCORES card, list variable numbers of those variables used as controls. For example, if controlling on variables 1, 2, and 4:

```
STA(T2)(P).
$C-B(1)(2)(4).
```

NOTE: If there is only 1 observation and parameter 4 is set to 1, then the mean, standard deviation and variance for that variable will be set to zero. If blanks are checked and standard scores are requested, those observations which have a blank will remain blank after calculation of standard scores.

V. Example (Use of Parameter 10)

```
//  EXEC SOUP
//SYSIN  DD  *
MAT.
MOV(C)(S1).
END P
COR(S1)(S2).
MAT.
TRA(S2)(S3).
END P
STA(S1)(P)()()()()()()()(S3).
END S
DATA(10)(10F1.0)
   .
   .
   .
   .
   .
   .
   .
   .
   .
   .
END#
/*
```

ANALYSIS OF VARIANCE PACKAGE

BALANOVA 5

A General Analysis of Variance Program for the IBM 7094

Paul Herzberg   August 1966

Statistical Service Unit Research, University of Illinois, Urbana

Table of Contents

BALANOVA 5

This version of BALANOVA 5 is unchanged as far as computations are concerned from the original program written by Paul Herzberg in August, 1966. See his write-up for explanation and references for the calculations.

Each observation (row of data) input to this program must be identified by a number for each factor including the replication factor. These numbers (which cannot be punched in I format) represent the levels of the corresponding factors and must precede the dependent variables. In the output produced by the program, each factor is given a unique letter name, beginning with A. Thus the first column of the input data corresponds to the levels of factor A which is described on the first factor specification card (see below). Each additional factor is given the next letter in the alphabet, and a corresponding factor specification card. The dependent variables follow the factor levels on the input data, and they are numbered one through the total number of dependent variables, in the output of the program.

On the program call card, the following parameters follow the program name, BALANOVA 5; with the first four parameters being required.

Parameter Number	Use or Meaning
1	Input Address. SEQUENTIAL 1-15 OR CARDS.
2	Number of factors counting replication factor if there is one. Maximum = 10.
3	Number of dependent variables. Maximum = 200.
4	Number of levels of the 1st factor.
5-13	Number of levels of the 2nd - 10th factors.
14	1 if desire unweighted means analysis even though have proportional cell frequencies.
15	1 to suppress printing of cell means.

Following the program card is a separate subparameter card (factor specification card) for each factor in the order in which the factors appear in the input data. Each card has the following parameters.

Parameter Number	Use or Meaning
1	0 if fixed factor 1 if random factor
2	0 if not the replication factor 1 if is the replication factor

Parameter Number	Use or Meaning
3-11	factors in which this factor is nested

As in other SOUPAC programs, parameters at the end of the card which are not used may be deleted and the period appear after the last non-zero parameter. The factor specification cards must be followed by an END PROGRAM card.

Chapter 1. General Description

1.1 Introduction

BALANOVA 5 is a general analysis of variance program applicable to a wide-range of balanced designs. In the case of designs with a replication factor, BALANOVA 5 allows inequality in the number of replications in each cell. If the number of replications is equal or proportional, the analysis is handled by least squares (weighted means). If the number of replications is not proportional then an unweighted means analysis is performed. This is an approximation to the least squares solution.

BALANOVA 5 accepts some designs that are not completely crossed, namely those nested designs in which all main factors are balanced. Hence hierarchical designs are allowed. As well, repeated measures designs are allowed. In these designs the replication factor is not nested in all the other factors.

The design model may be fixed-effects, random-effects or mixed. BALANOVA 5 automatically determines all the legal sources of variation (main effects and interactions) and determines the correct denominator mean square for those sources which can be tested by F test. In order to do this, BALANOVA 5 first generates the expected mean square table which is printed in readable form. The method used closely follows Scheffé (1959), Chapter 8.

BALANOVA 5 will accept most of the designs described in Winer (1962), Chapters 3, 4, 5, 6, and 7 and Lindquist (1953), Chapters 3, 5, 6, 6, 8, 9, 10, and 13 (Types I, III, VI). Chapter 2 and the Appendix of this manual contain a large number of examples drawn from these two books.

The author was somewhat reluctant to develop BALANOVA 5 and is hesitant to encourage its wide use for the following reasons:

1. A general program such as BALANOVA 5 encourages the use of statistics in a "cook-book" manner. Data is generated to fit the input specifications of the program with no consideration given to the theory of analysis of variance. The experimenter who uses a computer program in this way often neglects to consider whether the statistical test is appropriate for the work he is interested in and whether the assumptions needed for the test are satisfied in the particular experiment he has used.

2. In the author's experience, the results printed by such a program as BALANOVA 5 have a certain finality which encourages the user to accept the results as gospel truth. When used in this way, the experimenter forgets that there is a real possibility of programming or machine error.

3. In the particular case of analysis of variance, the idea has become widespread that the summary table of F ratios is the most important part of the analysis. This is not the case. The most important part of analysis of variance

is the estimation of the main effects and the interactions. Only by looking at their size can the experimenter evaluate what is happening in his experiment. In order to encourage this use of analysis of variance, BALANOVA 5 prints a table of marginal means which allows easy calculation of all the effects in the experiment. The F table is only a set of warning signals. A non-significant F indicates that the corresponding differences between effects can be attributed to sheer chance.

4. BALANOVA 5 performs an unweighted-means analysis when the replication numbers are non-proportional. The author fears that this option will be used too often and without consideration of its dangers. The unweighted-means solution is often not satisfactory and references on analysis of variance should be consulted. (Scheffé, 1959, Winer, 1962, Lindquist, 1953).

BALANOVA 5 was designed to reduce the great amount of hand computation needed in analysis of variance calculations. It was not intended to eliminate the necessity of the user being familiar with the theory of analysis of variance. It is hoped that the above comments will discourage some indiscriminant use of BALANOVA 5.

1.2 Special Features

The output from BALANOVA 5 consists of

1. A table of the expected mean squares in readable form.

2. The number of replications in each cell in the case of designs with a replication factor.

3. The table of marginal means. All means entering in the computation of the sum of squares are printed.

4. The analysis of variance summary table including, for each source of variation, the sum of squares and mean square, and for each source with denominator, the F ratio and the probability of the chance occurance of the F ratio.

A feature of BALANOVA 5 is its flexible specification of analysis of variance designs, allowing a wide range of designs to be described by a common code.

A large number of checks are made by BALANOVA 5 to ensure that the design is legal and that the data correspond to the design. Diagnostics are printed to indicate all error conditions.

1.3 Legal Design of BALANOVA 5

Consider the following definitions, taken from Scheffé (1959). Let there be p factors in a design, not counting the replication factor, if there is one. A cell is specified by a set of p levels, one for each factor. The layout of design is complete if there is at least one observation in every cell. The factors in such a design are completely crossed. If the design is complete and there is a replication factor (i.e. all cells have at least one observation and at least one cell has more than one observation) then the design is considered to be a Class A design in BALANOVA 5.

There are many analysis of variance designs which are not complete in the above sense. Examples of incomplete designs are Latin-square, incomplete blocks and nested designs. The only incomplete designs which are allowed in BALANOVA 5 are nested designs which are balanced in all factors except for the replication factor (which need not be balanced). These incomplete designs are called Class B and C designs. "Nesting", "balanced" and "replication factor" are defined in the nest three paragraphs. These definitions are illustrated in Chapter 2.

Nesting may be defined as follows: The levels of a factor C are nested within the levels of a factor A (in short, C is nested within A) if and only if each level of C appears with only a single level of A in the observations. Note that if C is not nested within A, it is crossed with A, but only if every level of C appears with every level of A is C completely crossed with A. Latin-square and incomplete block designs are only partly crossed.

A nested factor C is balanced if the number of levels of C is the same within each combination of those factors within which C is nested and the factors (if any) which are crossed with C are completely crossed.

A replication factor, in BALANOVA 5, is a factor which is nested within one or more other factors, but not necessarily within all other factors. Furthermore, no factor may be nested within the replication factor. That is, a factor is a replication factor if and only if for every other factor A in the design, it is either nested within A or crossed with A. A replication factor may be nested within some factors and crossed with others. There can be at most one replication factor in a design.

The distinction is made between replication factors and other nested factors in BALANOVA 5 since replication factors do not have to be balanced. All other factors must be balanced.

Using these definitions, the following designs are legal in BALANOVA 5.

Class A designs (completely crossed with nested replications)

Class A designs contain (p + 1) factors of which p are the main factors and the other factor is the replication factor. The following two conditions must both be met for the design to be Class A.

(a) All p main factors are completely crossed.

(b) The replication factor is nested in all main factors.

Thus one-way and factorial designs are Class A designs.

Class B designs (other replication designs)

Class B designs also contain (p + 1) factors of which p are the main factors and the other factor is the replication factor. However one or both of the two conditions, (a) and (b), are not satisfied in Class B designs.

When (a) is not satisfied, that is, the p main factors are not completely crossed, then the main factors must satisfy the following condition.

(a') Consider any two main factors, A and B. Either A is completely crossed with B, or A is nested within B or B is nested within A. This must be true for all pairs of main factors. Furthermore, at least one pair must have the nested relationship or else (a) would be satisfied.

When (b) is not satisfied, then the following condition must be true.

(b') The replication factor is nested in at least one but not all main factors. Note that the requirement that the replication factor be nested in at least one factor is part of the basic definition of a replication factor.

Class B designs then can be of the following two types.

Hierarchical designs: (a') and (b) are satisfied. The replication factor is nested in all factors but there is some nesting among the main factors.

Repeated measures designs: (b') is satisfied. Either (a) or (a') can be satisfied. The necessary feature (b') of repeated measures designs is that the replication factor is crossed with one or more of the main factors. The factors in which the replication factor is nested may themselves be either crossed (a) or nested (a').

Class C designs (no replication factor)

Class C designs have p factors and there is no replication factor. All factors must be balanced. For each pair of factors, e.g. factors A and B, either A is completely crossed with B, or A is nested within B or B is nested within A. There does not necessarily have to be any nesting at all.

In summary, then, designs are classed in the following way in BALANOVA 5. Class A and B designs have a replication factor, Class C designs do not. Class A designs are distinguished from Class B designs in that a Class A must have 1) all main factors completely crossed and 2) the replication factor nested in all main factors. Class B designs violate one or both these requirements.

In Class A and B designs, the replication factor does not need to be balanced. However all nested factors, except the replication factor (if any), must be balanced. Recall that in Class A designs, the replication factor is the only nested factor.

As explained above, the replication factor is distinquished from other nested factors since it does not have to be balanced. There are two other reasons for distinguishing the replication factor from other nested factors. These reasons are important even if the replication factor is balanced.

1. In Class A designs (completely crossed with replications) only cell means are stored in the computer and thus very large designs can be accommodated. The allowable number of replications in each cell is virtually unlimited.

2. For all replication designs, whether of Class A or B, the level number for the replication factor in each nest does not need to run from one up to the maximum number of levels in each nest as it does for all other factors. Any convenient numbering of the replications may be used (e.g. a unique number for every subject in the experiment, regardless of the nest within which he is). This feature of BALANOVA 5 is especially useful when several dependent variables are analyzed and there is missing data for some of the subjects for some of the dependent variables.

1.4 Calculations for equal and unequal number of replications

The calculations performed by BALANOVA 5 for designs with a replication factor (Class A and B designs) depend on whether the number of replications in each cell are equal or unequal. If the numbers are equal, the standard analysis of variance calculation is made (least-squares or weighted means analysis). If the numbers are unequal, a check is first made to see if the cell N's are proportional. In a two-way analysis of variance, for example, the cell N's are proportional if the number of replications in the ij cell, N_{ij}, satisfies

$$N_{ij} = \frac{N_{iT} \times N_{Tj}}{N_{TT}}$$

where the T's indicate marginal totals. If the cell N's are proportional, BALANOVA 5 makes the least-squares calculations, i.e. weighted means are used. If the cell N's are not proportional, then the method of unweighted means is used (See Scheffé, pp. 262-3 or Winer, pp. 222-4).

In general, if i, j, k,...., λ are those factors within which the replication factor is nested (not necessarily all the factors in the design), and if $N_{ijk....\lambda}$ is the number of replications in a particular nested cell, then the cell N's are proportional, if, for all combinations ijk....λ,

$$N_{ijk....\lambda} = \frac{N_{iTT....T} \times N_{TjT....T} \times \; . \; . \; . \; . \; \times N_{TTT....\lambda}}{(N_{TTT...T})^{q-1}}$$

In this formula, the T's indicate marginal totals and q is the number of factors within which the replication factor is nested. In particular, the one-way analysis with unequal N's is a proportional design (i.e. the cell N's are proportional) by this definition, since

$$N_i = \frac{N_i}{(N_T)^0} = \frac{N_i}{1} = N_i$$

In fact any design in which the replication factor is nested in only one factor is a proportional design.

1.5 Specification of a Design

Any design is described by listing the following information about each factor in the design, including the replication factor if there is one. The information for each factor is punched on a separate card (a factor specification card), and the cards should be in the same order as the factors are in the input data. Each parameter should be enclosed in parentheses, and each card terminated by a period.

Parameter 1

Type of factor. The first parameter on each factor specification card should be a zero if the factor is fixed, and a one if it is random. The replication factor is always a random factor. At least one factor in every design must be random.

Parameter 2

Replication Factor. If the design has a replication factor, this is indicated by punching a one for the second parameter. A design may have only one replication factor. If there is no replication factor, the second parameter should be zero (or blank) on all of the factor specification cards.

Parameter 3-11

Nesting. The factors in which the given factor (the one to which this card refers) is nested are listed. Factors are numbered from one through the number of factors in the design. If the factor is not nested, parameters 3-11 may be completely omitted.

An example of this way of specifying a design will be now given. Consider a two-way analysis of variance with subjects within cells. The design is considered to have <u>three</u> (not two) factors, namely A and B, the main factors, and C, the replication factor. Suppose there are 3 levels of A and 4 levels of B and that each cell has 10 subjects. The cards used to perform this analysis are listed below. Each line corresponds to one IBM card.

```
TRANSFORMATIONS(CARDS)(SEQUENTIAL 1)(4).
END PROGRAM
BALANOVA(SEQUENTIAL 1)(3)(1)(3)(4)(10).
(0 (0).
(0)(0).
(1)(1)(1)(2).
END PROGRAM
```

The first card listed above calls the TRANSFORMATION program and uses it to store raw data from cards onto sequential file number 1 (SEQUENTIAL 1). The number four is the total number of variables, independent (3) + dependent (1). Since no transformations are performed, the END PROGRAM card immediately follows the main program card.

The third card listed above calls the BALANOVA program. The first parameter is the location of the data (SEQUENTIAL 1), the second is the number of factors (3), the third is the number of dependent variables (1), the fourth is the number of levels of the first factor (3), followed by the number of levels of the second factor (4), and finally the number of levels of the last factor, which is the maximum cell size (10) when considering the replication factor.

The fourth card is the factor specification card for factor 1. The first 2 parameters are zero, labeling this factor as fixed, and as not being the replication factor. The fifth card is the factor specification card for factor 2 which is also fixed, and not the replication factor. Note that parameters 3-11 are blank, as factors 1 and 2 are not nested in any other factors.

The sixth card is the factor specification card for factor 3. Its four parameters denote it as a random factor, as the replication factor, and as nested in factors 1 and 2. The seventh card terminates the BALANOVA program.

Chapter 2. Design Examples

OMITTED TEMPORARILY

Chapter 3. Preparation of Input

3.1 Introduction

The following rules apply to the assignment of factor levels in all types of designs. There is usually no need to rekeypunch existing data however, as it is almost always possible to create the factor levels in the TRANSFORMATIONS program. If you need help using this program, see a SOUPAC consultant.

(a) Non-replication Factors in Class A, B and C designs

The levels for non-replication factors must run from one (1) consecutively up to the number of levels given on the BALANOVA call card. E.g., if a factor represents four treatment groups, these groups must be numbered 1, 2, 3, and 4 and each subject's row or rows in the data matrix must have a 1, 2, 3, or 4 punched to indicate the group he is in. If the factor is nested, the level numbers must run from one (1) up, in each cell of the nest. See the example given in Section 3.2.

(b) Replication Factors in Class A design

The replication numbers (level numbers) can be anything, for example, a subject identification number. The subject numbers do not have to be unique either in a group or between groups. In fact, to tell the truth, in Class A designs, the replication level is not used but it must nevertheless appear, even if it is a dummy. This statement does not apply to other design classes.

(c) Replication Factors in Class B designs, of repeated measures type

Special care must be taken with the replication levels in those designs. Let us divide the non-replication factors into two groups:

α-set: those factors in which the replication factor is nested.

β-set: all other factors - i.e. those factors crossed with the replication factor.

If the β-set is empty, the design is of hierarchical rather than repeated measures type. See paragraph (d) below.

Let us denote by an α-cell a particular set of levels of the factors in the α-set. The replications in this cell may be any values (not necessarily from one (1) up) but must be distinct. Again, the numberings in two different α-cells do not have to be distinct, but can be. In other words, if the replication factor is subjects, an identification number may be used as the replication level. Now each subject appears in more than one row (card) of the data matrix since each subject appears with every combination of levels of the factors in the β-set. Now it should be obvious that every row that refers to the same subject has to have the same replication number. This is the only way that BALANOVA 5 can tell that two different rows refer to the same subject.

(d) Replication Factors in Class B designs, of hierarchical type

The subject numbers must all be different within any one cell (one level set of the non-replication factors) but may be the same over different cells.

Special note on missing data

If a dependent variable field on a card is totally blank, BALANOVA 5 does not include the score in the analysis for the given dependent variable. However other non-blank dependent variable fields on the same card will be included in their respective analyses.

Do not confuse this deletion of missing data with an error comment by BALANOVA 5 to the effect that there is no data cell A = 1, B = 2. This comment means that no data card with A = 1, B = 2 had non-blank data for the given dependent variable.

3.2 Data matrix examples

Class A design

In all the following examples, it is assumed data is stored on sequential file number 1 (SEQUENTIAL 1).

Consider a two-way design with three subjects in each cell. For the purpose of BALANOVA, subjects are also considered to be a factor, the replication factor. Suppose that there are two dependent variables, and further that the factor specification cards are listed in the order given below, following the main program card.

```
BALANOVA(SEQUENTIAL 1)(3)(2)(2)(3)(3).
(0)(0).
(1)(1)(1)(3).
(0)(0).
END PROGRAM
```

Note that, contrary to the usual case, the replication is the second factor. This illustrates one flexible feature of BALANOVA 5. A data matrix could be:

1	1	1	20	19
1	2	1	8	8
1	3	1	4	4
1	4	2	-3	6
1	5	2	4	10
1	6	2	2	3
1	7	3	4	4
1	8	3	6	2
1	9	3	8	4
2	10	1	2	7
2	11	1	-4	8

```
2    12    1     25     2
2    13    2    126    15
2    14    2      2    20
2    15    2      3     3
2    16    3      4     4
2    17    3      5    -1
2    18    3      3     2
```

Note that the first column is the A level, the third column is the C level and the second column is the replication level, which in Class A designs can be anything. The last two columns are the dependent variables. Each row of the data matrix would be punched on one or more cards. A possible format would be (3F5.0,3X,2F6.0).

The order of the rows is immaterial. They could be in any order and have been written in a systematic order only for convenience.

Class B design - repeated measures

The data in Winer, Table 7.2-3 could be analyzed with the following program, using three factors and one dependent variable.

```
BALANOVA(SEQUENTIAL 1)(3)(1)(2)(4)(3).
(0)(0).
(0)(0).
(1)(1)(1).
END PROGRAM
```

Data Cards:

```
1   1   1   0
1   2   1   0
1   3   1   5
1   4   1   3
1   1   2   3
1   2   2   1
1   3   2   5
1   4   2   4
etc.
2   1   5   5
2   2   5   4
2   3   5   6
2   4   5   6
2   1   6   7
2   2   6   5
2   3   6   8
2   4   6   9
```

Again, the rows could be any order. The subjects in the second level of factor A could be assigned level numbers 1, 2, 3 or any other thre distinct numbers. A possible format for this matrix is (2F5.0,F6.0,1X,F7.0).

Another repeated measure design is Winer, Table 7.4-3. Here we have four factors and one dependent variable.

```
BALANOVA(SEQUENTIAL 1)(4)(1)(2)(2)(3)(4).
(0)(0).
(0)(0).
(1)(1)(1)(2).
(0)(0).
END PROGRAM
```

Data Cards:

```
1    1    1    1    18
1    1    1    2    14
1    1    1    3    12
1    1    1    4     6
1    1    2    1    19
1    1    2    2    12
1    1    2    3     8
1    1    2    4     4
etc.
1    2    6    1    18
1    2    6    2    10
1    2    6    3     5
1    2    6    4     1
2    1    7    1    16
2    1    7    2    10
2    1    7    3     8
2    1    7    4     4
etc.
2    2   12    1    16
2    2   12    2    12
2    2   12    3     8
2    2   12    4     8
```

The assignment of levels to factors A, B, D (columns 1, 2, 4) has to be as shown above, but the subjects numbers can be changed provided that, for each (A,B) cell, no two subjects have the same number. A possible format is (2F5.0, 1X,2F5.0,F12.0).

Class B design - hierarchical

Consider the following design with three factors and one dependent variable.

```
BALANOVA(SEQUENTIAL1)(3)(1)(2)(2)(4).
(0)(0).
(0)(0)(1).
(1)(1)(1)(2).
END PROGRAM
```

This is a hospitals (factor 2) within drugs (factor 1) design, illustrated below, with unequal cell size.

Data Cards:

```
    1    1    1    5.0
    1    1    2    4.2
    1    2    1    5.6
    1    2    2    3.2
    1    2    3    4.6
    2    1    1    5.3
    2    1    2    8.2
    2    1    3    4.3
    2    1    4    6.3
    2    2    1    5.7
    2    2    2    6.8
```

Note that the hospitals are numbered 1, 2 in each level of factor 1 even though there are four different hospitals involved. This is necessary since factor 2 is a non-replication factor - see the rules about Data Cards in Section 3.1. There are 2 patients in hospital 1 for drug 1, 3 patients in hospital 2 for drug 1, 4 patients in hospital 1 for drug 2 and 2 patients in hospital 2 for drug 2. The patient numbering is flexible - it could be a different number for every patient, regardless of hospital. A possible format is (3F5.0.F7.1).

Class C design

The example in Winer, Table 4.3-1, could be set up as follows, with two factors and one dependent variable.

```
    BALANOVA(SEQUENTIAL 1)(2)(1)(5)(4).
    (1)(0).
    (0)(0).
    END PROGRAM
```

Data Cards:

```
    1    1      30
    1    3      16
    2    1      14
    2    2      18
    2    3      10
    2    4      22
    1    2      28
    3    1      24
    3    2      20
    3    3      18
    3    4      30
    4    1      38
    1    4      34
    4    2      34
    4    3      20
    4    4      44
    5    4      30
    5    3      14
    5    2      28
    5    1      26
```

The rows have been written in a non-systematic order to emphasis that, without exception, in BALANOVA 5 the data rows can be in any order. A possible format is (2F5.0,F10.0).

Chapter 4. Program Details

4.1 Method and program flow

The program follows the procedures in Scheffé (1959), Chapter 8. Scheffé's discussion will not be repeated here, but only a general description of the program flow will be given. The names of the subroutines used are indicated in case reference is made to the program listing. Many of the minor steps and subroutines are not described.

1. Main Program

Calls subroutines and routes your data through the program.

2. Design input and check (INPUTD)

The Factor Specification Cards are read and checked for errors. Many of the error conditions mentioned in Section 4.3 are checked in INPUTD. The design is transformed into the symbolic notation of live, dead and absent subscripts as in Scheffé.

3. Derivation of all legal sources (LEGALS,NEWS)

All possible interactions are generated but only one interaction with a given set of subscripts is retained. The procedure is identical to Scheffé, p. 277. para. 1. The program now has a list of all legal sources (including the original factors).

4. Expected mean squares (AUXIL,EMS)

The expected mean squares for each source are, of course, not computable numbers, but rather symbolic expressions. (cf. last column, Table 8.2.2, Scheffe). The program generates and prints these expression in a form very close to the normal printed form. The method is from Scheffé, pp. 284-8.

5. Denominator for each source (FINDEN)

By the standard procedures, using the expected mean squares, the program determines the correct denominator (if any) for each source.

6. Sorting of sources for summary table (SORT)

The sources are sorted in a convenient order, combining all sources with the same denominator. This order is then used in printing the summary table.

7 Input of data (INPUTX)

The input data is read from the input device (the first parameter on the main program card). The grand means for each dependent variable are computed, ignoring missing (blank) data.

8. Storage of data for one dependent variable (READX)

This routine as well as all the remaining ones are executed in cycle once for each dependent variable. READX stored the data in core and checks that no data is missing in the design. The data is actually stored as deviations from the grand mean. This is done to improve accuracy. See Section 4.2.

9. Check of replication numbers (CELLN)

In the case of Class A and B designs, BALANOVA 5 checks whether the call frequencies are equal, proportional or non-proportional.

10. Computation of sum of squares (SSEQU,SSPROP,XMEAN)

The marginal means and sums of squares for each legal source are calculated.

11. Computation and printing of final summary table (FISHER,FPRINT)

These calculations are made in the standard way.

4.2 Some comments on accuracy of computation

An attempt was made in the design of this program to eliminate the largest sources of computational inaccuracies that can occur in analysis of variance calculations.

Consider a one-way analysis with the following data:

	Group 1	Group 2	Group 3
	8.77	8.88	8.96
	8.79	8.90	8.99
	8.80	8.90	9.00
	8.82	8.91	9.02
	8.82	8.91	9.03
Means	8.80	8.90	9.00

Sums of squares computations are generally made as the sums and differences of two or more terms. In this example, the exact calculations would be

$$\begin{aligned} \text{SS between} &= 1188.2500 - 1188.1500 \\ &= 0.1000 \end{aligned}$$

$$\begin{aligned} \text{SS within} &= 1188.2554 - 1188.2500 \\ &= 0.0054 \end{aligned}$$

Note that these answers each have a string of zeros following the given digits since they are exact. However on a computer, with about 8 digit accuracy, the differences would only be accurate to about four decimals due to the cancellation of all the higher order digits by subtraction.

This is illustrated by a calculation using the previous analysis of variance program in SOUPAC. SOUPAC's answers were:

SS between = 0.10000610 (5 significant digits)

SS within = 0.0053863525 (2 significant digits)

Note the large errors in these SS. Even worse errors can occur in other data.

The whole problem could be avoided by accumulating a true sum of squares, that is, by adding positive numbers to form each SS rather than taking a difference of two large numbers. However this procedure was rejected because it is extremely slow.

The following procedure is used in BALANOVA 5 and it is very effective. The data are internally transformed to deviations from the grand mean. This is why the grand means are computed in subroutine INPUTX before the deviations are actually stored in the memory in subroutine READX. When the deviations are used the individual terms which are added and subtracted to give each SS are now numbers of approximately the same size as the SS itself. This means that the number of significant digits in the SS is large even if the grand mean is large. In the example given above, the deviation scores are:

	Group 1	Group 2	Group 3
	-.13	-.02	+.06
	-.11	.00	+.09
	-.10	.00	+.10
	-.08	+.01	+.12
	-.08	+.01	+.12
Means	-.10	.00	+.10

and the SS are computed as

SS between = 0.10000000 - 0.00000000
= 0.10000000

SS within = 0.10540000 - 0.10000000
= 0.00540000

The actual results produced by BALANOVA 5 were

SS between = 0.099999998 (8 significant digits)

SS within = 0.0053999992 (7 significant digits)

Note the great improvement in accuracy.

As a final feature of BALANOVA 5, the approximate number of significant digits in each SS is calculated and printed alongside each SS. These numbers should not be interpreted exactly but only as a warning when they are small. The approximate number of significant digits is calculated in the following way:

(a) Find the largest term, in absolute value, entering into the calculation of the SS. In the example above, the largest term in SS within is the first term (0.10540000).

(b) Take the ratio of this largest term to the SS itself. In the example, this ratio = 0.10540000/0.00540000.

(c) The approximate number of significant digits is then = 8.0-$\log_{10}$ (ratio). In the example, this is 8.0 - 1.38 = 6.62 which is printed by BALANOVA 5 as 7, a pretty good estimate.

Note that the number of significant digits printed by BALANOVA 5 reflects the loss of accuracy in the computation of the SS from two or more terms. It does not reflect loss of accuracy due to computation of the terms themselves.

4.3 Error conditions

BALANOVA 5 makes a detailed check to insure that the design is legal, that none of the computer storage arrays are exceeded and that all the data corresponds to cells within the specified design. The following general types of errors are distinguished and corresponding error messages are printed giving detailed instructions about how to correct the error.

1. One of the restrictions on program size has been exceeded. These restrictions are:

(a) maximum number of factors = 10

(b) maximum number of legal sources = 100

(c) maximum size of X-storage array (used for data, means and cell numbers) = 10,000

(d) maximum number of dependent variables = 200

(e) maximum number of sigma-squared terms in any one expected mean square = 10

2. The factor specification cards are incorrect or inconsistent. This is, the design is illegal. The checks made are:

(a) all nested factors must be listed as a factor.

(b) no factor may be nested within itself.

(c) at most one factor can be the replication factor. Furthermore, the replication factor must be nested in at least one other factor and no factor can be nested in the replication factor.

(d) the factor type must be fixed or random.

(e) the maximum number of levels for each factor must be more than one.

(f) there must be at least one denominator term in the analysis of variance summary table. If this is not the case it is probably due to no factor being designated as a random factor.

3. A Data Card has a level set which exceeds the limits stated in the maximum number of levels on the Factor Specification Cards.

4. Once the data for a dependent variable has been read in, a detailed check is made to insure that all cells in the design are filled. If one cell is not, the calculation for that design is deleted and the program moves on to the next dependent variable after printing sufficient information for the user to locate the missing datum. An additional check for Class B and C designs is made to insure that data for a given subscript set is not read in twice. If two data cards specify the same level set, a comment is made to this effect and the calculations for the dependent variable are deleted. Note that both the checks mentioned in this paragraph are made independently for each dependent variable and are made after the missing data (blank fields) for that dependent variable have been deleted. Errors referred to in this paragraph are not fatal and the program proceeds to the next dependent variable.

5. About one dozen other checks are made. They should always be passed satisfactorily since the design is first checked as above. These additional checks were inserted to assist in debugging the program and if one of them fails it indicates a remaining error in the program. A printed message is made to this effect in these cases.

4.4 Program Checkout

BALANOVA 5 has been checked on a large number of designs. Among these, the following calculations were reproduced by BALANOVA 5:

1. Lindquist, p. 266, Class A.

2. Winer, Table 7.8-3 (p. 376), both least-squares and unweighted means, Class B.

4.5 Number of levels of the replication factor

The rules in Section 1.5 and Chapter 3 are strict in the sense that, if they are followed, BALANOVA 5 will execute correctly. However the rules may be relaxed or ignored in the case of the number of levels of the replication factor and it is sometimes convenient to do so.

In Class A designs, any number of levels of the replication factor may be punched on the Factor Specification Card provided the number is ≥ 2. This is so because in Class A designs only cell means are stored and the program does not check the replication number anyway. This rule relaxation is useful when it is inconvenient for the user to calculate ahead of time how many subjects are in each cell.

In Class B designs, any number of levels of the replication factor may be punched on the Factor Specification Card provided

(a) the number is $\geq$ the maximum number of replications in any one nest, and

(b) the number is not so large that the restriction on the size of the X matrix (10,000) is exceeded.

Again this rule relaxation saves the user from having to know the maximum number of replications before using BALANOVA 5, provided he knows an upper limit. The restriction on the size of the X matrix will not often be exceeded. However, the user has not been informed, in this manual, how to estimate the size of the X matrix needed, since this limit is complicated to specify.

4.6 Acknowledgements

The general theory of balanced analysis of variance designs was first introduced to the writer by Professor R. W. Frankmann. The procedures used in this program are very similar to the approach he taught.

The author is indebted to John Paraskevopoulos who helped greatly in the revision of this manual.

4.7 References

Hays, W. L.: Statistics for Psychologists. New York: Holt, Rinehart and Winston, 1963.

Lindquist, E. F.: Design and Analysis of Experiments in Psychology and Education. Boston: Houghton Mifflin, 1953.

Scheffé, H. A.: The Analysis of Variance. New York: Wiley, 1959.

Winer, B. J.: Statistical Principles in Experimental Design. New York: McGraw-Hill, 1962.

Appendix. Key to Designs in Winer and Lindquist

OMITTED TEMPORARILY

CLASSIFICATION

I. General Description

The CLASSIFICATION program is designed to measure individuals against previously determined groups in order to determine probable group membership. The classification is done in a reduced test space derived from discriminant analysis. The method of classification is based on the premise that a group is totally described by its mean (or centroid) and dispersion; the individual's relation to each group is determined by a X^2 which indicates how many members of the group are farther from the centroid than he, and a Bayesian probability of membership in the group based on this X^2. For each individual, the X^2 and probability for each group are given; the user then applies a decision rule of his choice for assigning individuals to groups.

Since analysis is to be performed in a reduced space, the means and dispersion of each group must also be reduced to this space. The CLASSIFICATION program performs these reductions.

The calculation of probabilities requires the specification of the number of members in each group against which the individual is being compared. This may be the numbers actually in the groups used for finding experimental means and standard deviations or the number of individuals from the total group being tested who are to be assigned to each group. These numbers are specified as subparamters.

In every case the input is expected in the form in which it is output by the DISCRIMINANT ANALYSIS program. Input is limited to 16 groups and 50 variables.

II. Parameters

The program name CLASSIFICATION is followed by these parameters on the program call card:

Parameter Number	Use or Meaning
1	Input Address of discriminant vectors. The vectors are expected as columns. CARDS or SEQUENTIAL 1-15.
2	Input Address of means. The means for a given group are expected as a row. CARDS or SEQUENTIAL 1-15.
3	Input Address of dispersion matrices. The dispersion matrices are expected in a vertically augmented form. CARDS or SEQUENTIAL 1-15.
4	Input Address of individual scores. CARDS or SEQUENTIAL 1-15.

Parameter Number	Use or Meaning
5	Output Address for inverse of dispersion matrix. PRINT or SEQUENTIAL 1-15.
6	Output Address for X^2 and probabilities for each subject. SEQUENTIAL 1-15 and/or PRINT. (Output on SEQUENTIAL is in the form $N_i, X_{i1}, \ldots\ldots X_{in}, Y_{i1}, \ldots\ldots Y_{in}$ where N_i = sequential subject number in groups $X_{ij} = j^{th}$ probabilities for i^{th} subject $Y_{ij} = j^{th}$ X^2 for i^{th} subject)
7	Number of groups.
8	Number of variables.
9	Input Address of group sample sizes. CARDS or SEQUENTIAL 1-15.*

* Must be a row vector (1 observation) with the number of discriminant vectors as the first variable. The form output by Discriminant Analysis Program may be used. If provided independent of Discriminant Analysis, be sure to use single precision integer mode (I-format, fixed mode). If sample sizes and any other input are from cards, sample sizes are given first.

DISCRIMINANT ANALYSIS

I. GENERAL DESCRIPTION

Suppose that we have k populations (groups) and p measures (variables) on each member of each population. We want to test the hypothesis that our groups are significantly different on the entire set of variables. This one-way multivariate analysis of variance hypothesis is tested by this program. The program then locates the dimensions (discriminant functions) along which the group differences are maximum. Thus, we need some function to transform the p variates into a smaller set of independent measures which will indicate the differences between the groups. The DISCRIMINANT ANALYSIS program finds the independent linear functions of the variables which maximally discriminate between the populations (groups input). The results from this program, namely the discriminant functions, may be used in the CLASSIFICATION program to determine the probability that any subject belongs in any group. Also, by looking at the coefficients of the functions, we can determine to what extent each of the p variates contributes to each function. In order to do this we need to determine the coefficients of the functions such that the ratio of variances between groups to the variances within groups is maximized, i.e. the differences between groups are to be large relative to the differences within groups.

In matrix terms, we are trying to maximize the ratio

$$\lambda i = \frac{f_i' A f_i}{f_i' W f_i}$$

where f_i is the eigenvector associated with the i^{th} eigenvalue λi of $W^{-1} A$, A = the covariance matrix between means,

$$a_{ij} = \sum_{g=1}^{k} N_g (\overline{X}_{ig} - \overline{X}_i)(\overline{X}_{jg} - \overline{X}_j)$$

and W = the covariance matrix within classes,

$$w_{ij} = \sum_{g=1}^{k} [\sum_{n=1}^{N_g} (X_{ign} - \overline{X}_{ig})(X_{jgn} - \overline{X}_{jg})]$$

where k = number of groups, N_g= number of subjects in group g, N = total number of subjects, and i and j run from 1 to p, where p = number of variables.

To find the maximum, we derive from the partial derivatives of that ratio, the matrix equation

$$(W^{-1} A - \lambda I) F = 0$$

where F is the matrix of eigenvectors. The eigenvectors are the coefficients of the discriminant functions. The relative sizes of the eigenvalues indicate the extent to which the associated discriminant functions distinguish among the groups. The percentage of the total discriminating power of the variables contained in the jth discriminant function is represented by

$$100 \left(\frac{\lambda_j}{\sum_{i=1}^{N} \lambda_i} \right)$$

(N should be the smaller of k-1 and p)

In addition to obtaining the eigenvalues and discriminating coefficients, the program will compute scaled vectors to show the relative contributions of the variables to the discriminant function by

$$f_{ij}{}' = (w_{ii})^{1/2}/f_{ij}$$

II. INPUT

Input to the DISCRIMINANT ANALYSIS program consists of two or more data groups. Each data group consists of a set of observations on two or more variables. All of the groups must contain observations on the same set of variables. The groups may be input as separate card decks (each preceded by a DATA format card and followed by an END# card), as data groups located on separate temporary storage areas, or as a mixture of data groups on card decks and data decks on temporary storage areas. For examples see Section VIII.

III. SIGNIFICANCE TESTS

The measure of significance calculated in the DISCRIMINANT ANALYSIS program is a Wilks' lambda (likelihood ratio test statistic). This is a test of the discriminating power of the test battery. It tests the hypothesis that the population centroids (mean vectors) are equal for the k groups. The Wilks' lambda is a function of the roots of $W^{-1}A$ and is of the following form:

$$\Lambda = \prod_{i-1}^{r} \frac{1}{1-\lambda_i}$$

where r is the lesser of k - 1 and p. In matrix terms this criterion is defined in the following manner:

$$\Lambda = \frac{|W|}{|T|}$$

where W is the pooled within groups deviation score cross-products and T is the total sample covariance matrix. As |T| increases relative to |W| the ratio decreases in value with an accompanying increase in the confidence that the group centroids are not equal.

An F ratio which yields an approximate test of the significance of the Wilks' lambda is calculated and printed.

$$F = \left(\frac{1 - y}{y}\right)\left(\frac{ms + 2\lambda}{2r}\right)$$

where $s = \sqrt{(p^2q^2 - 4)/(p^2 + q^2 - 5)}$ $\qquad q = k - 1$

$m = n - (p + q + 1)/2$ $\qquad n = N - 1$

$\lambda = -(pq - 2)/4$ $\qquad N$ = total number of subjects

$r = pq/2$ $\qquad k$ = number of groups

$y = \Lambda^{1/2}$ $\qquad p$ = number of variables

The degrees of freedom to be used with the F value printed in the output are printed and are labeled F1 (degrees of freedom for the numerator) and F2 (degrees of freedom for the denominator) and equal 2r and mx + 2λ.

IV. OUTPUT

The output consists of the following:

1. Means of input variables for each group and group sample size (Parameter Number 8)

2. A dispersion matrix for each group. (Parameter Number 9)

3. The total sample deviation score cross-products matrix

$$t_{ij} = \sum_{n=1}^{N} (X_{in} - \overline{X}_i)(X_{jn} - \overline{X}_j)$$

   where i and j range over the variables. This matrix is the sum of the A and W matrix described in section I. This is the T matrix referred to in Parameter Number 7. The diagonal of this matrix contains the sums of squares. (Parameter Number 6)

4. The pooled within-groups deviation scores cross-products matrix which is labeled W on the output. (Parameter Number 6)

5. The total number of subjects in all the groups combined. (Parameter Number 6)

6. The means and standard deviations of the variables across all groups. (Parameter Number 6)

7. The correlation matrix of variables over all groups. (Parameter Number 6)

8. The among groups cross-products of deviations of group from grand means weighted by group sizes. This matrix is labeled A matrix on the output. (Parameter Number 6)

9. The eigenvalues for the W^{-1} A matrix. (Printed)

10. The eigenvalues and percentage of variance explained by each additional eigenvalue. (Printed on output, automatically)

11. The trace of the W^{-1} A matrix. This is the sum of the eigenvalues. (Printed on output, automatically)

12. The discriminant functions (f_{ij}). The number of discriminant function will equal r where r is the lesser of the two values k-1 and p, where k = the number of groups and p = number of variables. (Parameter Number 1)

13. The group means on the discriminant functions. This is a k x r matrix formed by multiplying the group means on variables and the discriminant functions. The matrix may be used to determine the relative positions of the groups on the derived function. (Parameter Number 11)

14. The scaled vectors. These vectors are formed by multiplying the discriminant functions by the square roots of the diagonal of the W matrix described above. The scaled vectors show the relative contributions of the input variables to each of the discriminant functions. (Parameter Number 5)

15. The measures of significance described in Section III. (Parameter Number 4)

V. RESTRICTIONS

The DISCRIMINANT ANALYSIS program follow the program name on the main program card. Each parameter must be enclosed in parentheses. The parameters must appear in the order given below. If a parameter is not needed, do not punch anything between its parentheses. All parentheses after the last non-empty pair may be omitted.

Parameter Number	Use or Meaning
1	Ω Output address of discriminant functions (Matrix f_{ij}). SEQUENTIAL 1 -15 and/or PRINT. (Needed for CLASSIFICATION).
2.	Number of variables
3	Number of groups
4	1 if desire significance measures printed.

Parameter Number	Use or Meaning
5	1 if desire scaled discriminant vectors printed.
6	1 if desire intermediate results printed.
7	1 if input is W and T matrices instead of raw data.
8	Output Address of group means on original variable and sample size (printed only).* SEQUENTIAL 1-15 and/or PRINT. (See Parameter 12) (Needed for CLASSIFICATION).
9	Ω Output Address of group dispersion matrices of original variables.* SEQUENTIAL 1-15 and/or PRINT. (Needed for CLASSIFICATION).
10	N - total number of subjects inall groups combined. This parameter is left blank if raw data is input rather than W and T matrices.
11	Ω Output Address of group means on discriminant functions. SEQUENTIAL 1 -15 and/or PRINT.
12	Output Address of group sample sizes. Possibly needed for CLASSIFICATION.

Input addresses of raw data input groups or W and T matrices are listed on a $INPUT card. The number of raw data groups is limited to 16. W precedes T in the sequence when these are input. W and T may be output using a $OUTPUT card. Output order is the same as input.

* If W and T are input instead of raw data, group means and dispersion matrices are not printed. Means and dispersion matrices on discriminant functions are not computed in this case.

Ω It is possible to print in F format and/or punch the output from these parameters. If you need either of these options, see the section in the INTRODUCTION on INPUT and OUTPUT.

VII. SPECIAL COMMENTS

1. This program does not check for missing data. All blank spaces are read as zeros.

2. The user is cautioned against using the DISCRIMINANT ANALYSIS program without an understanding of the statistical technique used. See the references (Section IX).

VIII. EXAMPLES

A

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
MATRIX.
MOV(C)(S5).
END P
DIS(S1)(40)(4)(1)()()()(32/P).
$INP(C)(C)(S5)(C).
END SOUPAC
DATA(40)(40F2.0)
   .
   .
   .
   .
   .
END#
/*
```

B

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
DIS(S1/P)(33)(4)(1)( )( )(1).
$INPUT(C)(C).
END SOUP
DATA(33)(33F2.0)
.
.    (Data for W)
END#
DATA(33)(33F2.0)
.
.    (Data for T)
END#
/*
```

In example A , four groups of data are being input with the first two groups coming from cards, the third from temporary storage on S5 and the fourth from cards. Discriminant functions are stored on S1 and printed and group means are stored on S2 and printed. Significance measures are calculated for 33 variables input. In example B, W and T matrices are input and much the same results are obtained as for example A. Group means cannot be calculated.

```
/*ID   (accounting information)
//  EXEC  SOUPAC
//SYSIN  DD  *
DIS(S1)(15)(2)()()()()(S2)(S3)()()(S4).
$INP(C)(C).
CLA(S1)(S2)(S3)(C)(P)(P)(2)(15)(1)(S4).
END S
DATA(15)
   .
   .
   .
   .
   .
END#
/*
```

This illustrates the use of the DISCRIMINANT and CLASSIFICATION programs. The DISCRIMINANT program will save discriminant functions on S1; it will operate on 15 variables for each of two groups. It will output group means on S2 and groups dispersion matrices and store group sample sizes on S4.

CLASSIFICATION in turn will read discriminant functions, group means, and group dispersion matrices from S1, S2, and S3 respectively. It will read the group to be classified from cards and print inverse of dispersion matrices and x^2 and probability. It will expect two sets of group means and dispersion matrices from 15 variables but 1 discriminant function (See Section IV). Original group sample sizes are read from S4.

IX. REFERENCES

Bryan, J. F., The Generalized Discriminant Function: Mathematical Foundation and Computational Routine. Harvard Educational Review, (1951) 21:90-95.

Cooley, W. W. and Lohnes, P. R., Multivariate Procedures for the Behavioral Sciences. New York, Wiley, 1962.

Kendall, M. G., A Course in Multivariate Analysis. New York, Hafner, 1961, pp. 106-108.

References on output:

Wilks' Lambda - first formula, page 119, Cooley and Lohnes.
F ratio - formula 4.1, page 62. Cooley and Lohnes.

T-TEST

I. General Description

The T-TEST program calculates a T coefficient or F ratio as described below:

Suboperation (1): Paired T-Test (also called correlated T-Test). Variables are in a row, variable on is paired with variable two, three with four, etc., and a paired T coefficient is calculated for each pair as follows:

$$t = \bar{d}/s_{\bar{d}}$$

$$\bar{d} = \sum_{i=1}^{N} [X\sigma - X\beta]/N_j$$, where σ and β are the j^{th} pair of variables

$$S_{\bar{d}} = N_j \sum_{i=1}^{N_j} [X\sigma - X\beta]^2 - (\sum_{i=1}^{N} [X\sigma - X\beta])^2/N_j^{2}.f]^{1/2}$$

where f = degrees of freedom = N_j or N_j -1 as desired and N_j is the sample size for the j^{th} pair of variables.

Suboperation (2): Paired T-Test for all possible combinations of variables computed as in Suboperation (1).

Suboperation (3): Test of differences from a known population mean. A population mean must be provided for each column of data or the mean will be set to zero. Population means should be provided as a row vector. The following are calculated and printed for each variable:

t value: $t = (\bar{X} - \mu)/S_{\bar{x}}$

where μ = parameterized value or zero

Mean: $\bar{X} = \sum_{i=1}^{N_i} X_i/N_i$

Standard Deviation: $S.D. = (\frac{N_i \Sigma X_i^2 - (\Sigma X_i)^2}{N_i x(d.f.)_i})$

NOTE: N-1 is the usual degrees of freedom, but N may be specified.

Standard Error of Mean: $S_{\bar{x}} = \frac{S.D.}{\sqrt{N}}$

Suboperation (4): Test of differences from a known population mean for previously analyzed data: the mean ($\bar{X}$), standard deviation (S. D.), and sample size (N) as well as the population mean [see suboperation (3)] ar read in and the program computes the standard error of the mean and T for each trio of the X, S.D., and N read in that order. If R trios of data are used (R observations on rows occur) then R population means should be given. Calculations are the same as in Suboperation (3).

NOTE: A column of data of Suboperation (3) is reduced to a three item row here.

Suboperation (5): Test of differences between two or more group means taken pairwise: each group is located on a separate storage location or set of cards. The following are calculated:

t value: $t = (\bar{X}_i - \bar{X}_j)/S_{\bar{x}_i - \bar{x}_j}$

Mean: $\bar{X}_i \quad \sum_{j=1}^{N_i} X_j/N_i$

Standard Deviation: $S.D. = \left(\frac{N_i \Sigma_i X_i^2 - (\sum_{i=1}^{N} X_i)^2}{N_i x (d.f.)_i}\right)^{1/2}$

Pooled Estimate of Variance: $S^2 = \left[\frac{N_i \sum_{i=1}^{N} X^2 - (\sum_{i=1}^{N_i} X^2)}{N_i} + \frac{N_j \sum_{j=1}^{N_j} X^2 - (\sum_{j=1}^{N_j} X^2)}{N_j}\right](d.f.$

Estimate of Standard Error: $S_{\bar{x}_i - \bar{x}_j} = \left[\frac{S^2(N_i + N_j)}{N_i N_j}\right]^{1/2}$

Suboperation (6): One-way analysis of variance. The data to be compared are located on different storage units or sets of cards. Calculations are made as follows for each variable:

S = number of storage units = number of subroups

N_i = number of observations in i^{th} subgroup, $i=1,\ldots\ldots\ldots, S$, and for each i, $j=1,\ldots\ldots\ldots, N_i$

X_{ij} = element in the j^{th} row of the i^{th} subgroup

$N = \sum_{i=1}^{S} N_i$ = total observations

$$\text{Total SS} = \sum_{i=1}^{S} (\sum_{j=1}^{N_i} X_{ij}^2) - (\sum_{i=1}^{S} \sum_{j=1}^{N_i} X_{ij})^2/N$$

$\sum_{j=1}^{N_i} X_{\cdot j}$ = sum of each constant i (i.e. over the subgroup)

$$\text{Between SS} = \Big[\sum_{i=1}^{S} \sum_{j=1}^{N} X._j{}^2 \Big]/N_i - \Big(\sum_{i=1}^{S} \sum_{j=1}^{N} X_{ij} \Big)^{2/N}$$

Within SS = Total SS - Between SS

Within D.F. = Total D. F. - Between D. F.

$$F = \frac{\underline{\underline{BSS}}\,/\,\underline{\underline{BDF}}}{\underline{WSS}\,/\,\underline{WDF}}$$

Suboperation (7) : One way analysis of covariance. The experimental (dependent) variable comes first followed by up to 49 covariates. The dependent variables is adjusted to the set of covariates, not iteratively to one covariate at a time. Subgroups or factor levels are handled as in analysis of variance, Suboperation (6), i.e. as separate input decks or temporary storage locations. Coefficients obtained are means and standard deviations, test of homogenity of regression, F-ratio for covariance, and adjustment coefficients. For further discussion see Winer, p. 578 ff.

References

Bryant, E. C., Statistical Analysis. New York: McGraw-Hill, 1960.

Snedecor, G. W., Statistical Methods. Ames: Iowa State College Press, 1957.

Winer, B. J., Statistical Principles in Experimental Design. New York: McGraw-Hill, 1962.

II. Restrictions

The maximum number of input variables is 150 except for Suboperation (2), all possible paris, and Suboperation (7), covariance, where the maximum is 50. Suboperations (5) and (6), tests of differences and analysis of variance, require the data to be divided into 2 or more subgroups. The maximum number of subgroups is 14.

III. Parameters

T-TEST and ANALYSIS OF VARIANCE

Parameter Number	Use or Meaning
1	Suboperations 1 - 7. (See above).
2	Number of subgroups (if applicable).

Parameter Number	Use or Meaning
3	0 - Count clanks as zeros 1 - Count blanks as missing data
4	0 - use N-1 as degrees of freedom 1 - use N as degrees of freedom (See Special Comments).

Input addresses of subgroups for options 5, 6, and 7 are listed on a $-INP card. (See section on SOUPAC Input/Output). The maximum number of subgroups is 14. See example for illustration. If options 1 or 2 are used, provide only 1 input address. For option 3, provide two addresses, the first for the sample being tested, the second for the criterion means. For option 4 provide two input addresses, the first for the sample being tested, the second for the criterion means, standard deviations and sample size.

IV. Special Comments

If it is desired to use the option of comparing the mean with some population mean other than zero as in Suboperations (3) and (4), a row vector of μ must be included in temporary storage. This row vector must be of length N = number of variables.

Most work requires N-1 degrees of freedom.

V. Examples

```
T-TEST (1).
$INP(S1).
```

Paired T-Test on data stored on S1.

```
T-TEST (2)( )(1)(1).
$INP(S15).
```

Paired T-Test on data stored on S15 doing test on all possible pairs checking for missing data and using N degrees of freedom.

```
T-TEST (3)
$INP(S1)(S3).
```

T-Test of population means on S1 against criterion means on S3.

```
T-TEST (5) (2).
$INP(S1)(S2)
```

T-Test of group mean of two groups, one on S1 and the other on S2.

```
 T-TEST (6) (4) (1).
$INP(S1)(C)(C)(S3).
```

One-way analysis of variance over four groups, one on S1, two from cards, and one from S3, checking for missing data.

CORRELATIONS AND REGRESSION PACKAGE

BISERIAL CORRELATION

I. General Description

This program calculates the following coefficients for each combination of one dichotomous and one continuous variable.

Case totals: % cases in p = N_p/N

% cases in q = N_q/N

Total cases = N

Mean:

$$\bar{X}_p = \Sigma X_p/N_p$$
$$\bar{X}_q = \Sigma X_q/N_q$$
$$\bar{X} = \Sigma X/N$$

Standard deviation:

$$S_p^2 = \Sigma X_p^2/N_p - \bar{X}_p^2$$
$$S_q^2 = \Sigma X_q^2/N_q$$
$$S^2 = \Sigma X^2/N - \bar{X}^2$$

Biserial r:

$$r = \frac{(X_p - X_q)\ pq}{S\ (.3989)\ h}$$

where p = percentage of cases in 0 category
q = percentage of cases in 1 category
h = height of the normal curve computed from normal tables

The program checks for missing data, and computes the above measures only for those cases where both dichotomous and continuous variables are present.

II. Restrictions

A. Dimension

The input for the BISERIAL R program is limited to a maximum of 20 dichotomous variables and 100 continuous variables.

B. Type of input

Data is read row-wise from either tape or cards. All dichotomous variables must be first in each row. They should be coded with 0 and 1

III. Parameters

The program call card requires 4 parameters after the program name, BISERAL R:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15.
2	Output Address of correlations. SEQUENTIAL 1-15 and/or PRINT.
3	Number of dichotomous variables.
4	Number of continuous variables.
5	0 - ignore blanks 1 - count blanks as zeros

CANONICAL ANALYSIS

I. GENERAL DESCRIPTION

A. Mathematics

The Canonical Correlation program provides a multivariate test of the hypothesis that two sets of normally distributed variables are independent. The larger set of variables (the predictor variables) is considered to have q members, and the smaller set (criterion variables) has p members. This program also linearly transform each set of variables into a new set of independent variables, (or dimensions) such that the first new predictor (a linear combination of the original predictors) has maximum correlation with the first new criterion variable. The second new predictor is maximally correlated with the second new criterion, and so on (with the constraint that each new variable is uncorrelated with the previous new variables derived from the same set of original variables).

Let the criteria set consist of the p̲ variables $x_1,......,x_p$ and the predictor of q̲ other variates $x_{p + 1},....,x_{p + q}$. Assume $p \leq q$. We then look for weighting matrices $c_{d\beta}$ and d_{ab} such that

$$\xi_\alpha = \sum_\beta c_\beta \, x_\beta \qquad \begin{array}{l} \alpha = 1, 2, \ldots p \\ \beta = 1, 2, \ldots p \end{array}$$

$$\eta_a \quad \sum_b d_{ab} x_b \qquad \begin{array}{l} a = 1, 2, \ldots p \\ b = p+1, p+2, \ldots p+q \end{array}$$

The variables ξ and η have the following properties:

1. They are standardized variables.

2. Within each set, the ξ's are independent and the η's are dependent

3. Within the set ξ_α (as $\underline{\alpha}$ runs from 1 through p) and within the set of ηa (as a̲ runs from 1 through p) the correlation is zero. These matrices are formed in such a way that ξ_1 and η_1 , ξ_2 and η_2 , etc. are maximally correlated giving p correlations 1,....,p.

The purpose of this program is to find the p correlations $\lambda_1,...,\lambda_p$ and the weighting matrices $c_{\alpha\beta}$ and d_{ab}.

B Procedure

The correlation matrix R is first partioned into:

$$\begin{array}{c|c} A & C' \\ \hline C & B \end{array}$$

where A = correlation among predictors
C = correlation between predictors and criteria
B = correlation among criteria

The matrix equation

$$(CA^{-1}\ C' - \lambda^{2}B)\ U = 0$$

is solved to find the p correlations $\lambda_1, \ldots\ldots \lambda_p$, and the criteria weighting matrix U whose elements are $c_{\alpha\beta}$. The eigenvalues D and eigenvectors H of B are found: $CA^{-1}C'$ is then post-multiplied by $HD^{-1/2}$ and pre-multiplied by its transpose to reduce the equation to the standard form $(Z - \lambda I)U=0$. Then the predictor weighting matrix V whose elements are d_{ab} is computed by the equation

$$V = (A^{-1}\ C'\ U)\ ^{1}/_{\lambda}$$

II. INPUT

Input to the CANONICAL ANALYSIS program consists of a correlation matrix. These variables include a set of predictor variables and a set of criterion variables. Either set may be first on the input data but there can be no mixing of the two types of variables on the input data. The TRANSFORMATION program may be used to reorder the variables if they are mixed on the card data deck.

III. SIGNIFICANCE TESTS

Included in the printed output of the CANONICAL program is a Chi-square value for each of the eigenvalues λ^2 computed in the program. The chi-square values printed are determined from the Wilks' lambda values using the procedure outlined by Bartlett (see Section IX). The chi-square values provide a test of the null hypothesis that the p variates are unrelated to the q variates. If there is at least one way in which a linear combination of the predictor variables is correlated with a linear combination of the criterion variables this Chi-square value will be significant. The second Chi-square may then be examined. This Chi-square is a test of a second relationship after the first relationship has been removed. If this Chi-square is significant a second linear combination of the predictor variables is correlated with a second linear combination of the criterion variables. This process continues until the first non-significant Chi-square is found. All Chi-squares beyond that point will be non-significant.

IV. OUTPUT

The output consists of the following:

1. The matrix of standardized regression coefficients. This is the matrix of coefficients which would be formed if the raw data used to calculate the correlation matrix input had been converted to standard scores. The predictor variables are on the rows of the matrix and the criterion variables are on the columns of the matrix.

2. A multiple correlation squared (R^2) for each of the criterion variables. The first R^2 value is the multiple correlation of the first criterion

variable with the entire set of predictors variables. The second is for the second criterion variable with the set of predictors, etc.

3. A set of eigenvalues λ^2, correlations λ, Wilks' lambdas, Chi-squares, and degrees of freedom. (See Section III) (Printed)

4. A matrix of criterion weights. (Parameter Number 3)

5. A matrix of predictor weights. (Parameter Number 4)

V. RESTRICTIONS

A. Dimension
The maximum permissible number of variables in each set is 80.

B. Special Conditions
The matrices A and B must both be non-singular

C. Number of criteria (p) must be equal to less than the number of predictors (q).

VI. PARAMETERS

The parameters for the CANONICAL ANALYSIS program follow the program name on the main program card. Each parameter must be enclosed in parentheses. The parameters must appear in the order given below. If a parameter is not needed, do not punch anything between its parentheses. All parentheses after the last non-empty pair may be omitted.

Parameter Number		Use or Meaning
1		Input Address (correlation matrix). CARDS or SEQUENTIAL 1 - 5.
2		Output Address of canonical correlations. SEQUENTIAL 1 - 5 and/or PRINT.
3	Ω	Output Address of criterion weighting matrix. SEQUENTIAL 1 - 5 and/or PRINT.
4	Ω	Output Address of predictor weighting matrix. SEQUENTIAL 1 - 5 and/or PRINT.
5		Number of predictor variables.
6		Number of criterion variables. (Must be less than or equal to number of predictors).
7		Order of variable sets on input: 1 if predictors are first 2 if criteria are first

Parameter Numbers	Use or Meaning
8	Operation to be performed: 1 if only canonical correlations 2 if weights are to be computed
9	1 if want regression coefficients printed
10	1 if want multiple correlation squared (R^2) printed

Ω It is possible to print in F format and/or punch the output from these parameters. If you need either of these options, see the section in the Introduction on Input and Output.

VII. SPECIAL COMMENTS

This program does not check for missing data. All blank spaces are read as zeros.

VIII. EXAMPLES

1A

```
/*ID  <accounting information>

//  EXEC  SOUP

//SOUP.SYSIN  DD  *

CANONICAL (CARDS)(SEQUENTIAL 1/PRINT)

(PRINT)(PRINT)(15)(5)(2)(1).

ENDS

DATA
  .
  .
  .
  .
END#

/*
```

1B

```
/*ID <accounting information>

//  EXEC  SOUP

//SYSIN  DD  *

CAN (C) (S1/P)(P)(P)(15)(5)(2)(1).

ENDS

DATA
  .
  .
  .
  .
END#

/*
```

Examples 1A and 1B illustrate the use of the program call card for a CANONICAL correlation program. In these examples the input data consists of 15 predictor and 5 criterion variables and will be a card deck. The predictor variables are the first 15 variables of each observation in the data deck. The printed output will be the canonical correlations, the criterion weighting matrix, and the predictor weighting matrix. The canonical correlation matrix will also be stored on temporary storage SEQUENTIAL 1. 1A & 1B perform same calculations.

IX. REFERENCES

For a discussion of the uses of canonical analysis, see Kendall, M.G., A Course in Multivariate Analysis, New York, Hafner, 1961, pp. 68-85 or Kendall, M.G., The Advanced Theory of Statistics, New York, Hafner 1951, Vol. II., pp. 348-358.

For the derivation of the method used, see Anderson, T. W., An Introduction to Multivariate Statistics, New York, Wiley, 1958, pp. 288-296.

For significance test procedure, see Cooley, W. W. and Lohnes, P. R., Multivariate Procedures for the Behavioral Sciences, New York, Wiley, 1962, p. 37.

CORRELATION

I. General Description

The main purpose of the CORRELATION program is the calculation of Pearson product-moment correlations (hereafter referred to as correlations in this writeup). A correlation measures the linear dependency between two variables, and this program calculates a correlation for each pair of input variables. The square of a correlation, sometimes called the coefficient of determination, represents the proportional reduction in variance of one variable due to a linear relationship with another. Thus the coefficient of determination measures the strength of a linear relationship, or the proportion of variance accounted for by a linear rule.

The CORRELATION program automatically produces other types of correlation coefficients, because the calculations required are identical. Thus point biserial coefficients of correlation (often preferred to biserial correlation), phi coefficients (alternative to tetrachoric coefficients), and Spearman's rank order correlations can be readily obtained. Thus, if the input consists of dichotomous variables, the output will contain a mixture of phi's, point biserials, and ordinary correlations. (A point biserial correlation is a correlation between a dichotomous variable and a continuous variable). If the input to the correlation program consists of rank ordered data (ordinal), the output will be Spearman's rank order correlations. (See Walker and Lev, Chapter 11 for comparisons and comments on the above mentioned coefficients).

In the process of calculating the correlations, the means and standard deviations of the individual variables are computed, as are the cross-products and covariances between variables. After the correlations have been calculated, they are used to calculate the linear regression coefficient and corresponding intercept terms needed for predicting each variable from each other variable.

II. Input

Input to the CORRELATION program consists of a set of independent observations on two or more variables. The data is considered as a two-dimensional array (or matrix) of numbers with each column containing the observations on one variable, and each row consisting of one observation on each variable. If we use the letter X to represent the matrix of raw data, we let X_{ij} represent the i^{th} row (where $i = 1, 2, \ldots N$) and the j^{th} column (where $j = 1, 2, \ldots M$). In other words, we have N observations (rows) and M variables (columns) in our data matrix X.

III. Formulas and Calculations

The following formulas define certain statistics and illustrate their methods of calculating within the program. The subscript i refers to observations (or individuals) and runs from 1 to N. The subscripts j and k refer to variables, and they run from 1 to M.

Mean (of variable j)

$$= \bar{X}_j = \frac{\sum_{i=1}^{N} X_{ij}}{N}$$

Covariance* (between variables j and k)

$$= C_{jk} = \frac{\sum_{i=1}^{N} (X_{ij} - \bar{X}_j)(X_{ik} - \bar{X}_k)}{N - 1} = \frac{N \sum_{i=1}^{N} X_{ij} X_{ik} - (\sum_{i=1}^{N} X_{ij})(\sum_{i=1}^{N} X_{ik})}{N(N - 1)}$$

Standard Deviation* (of variable j)

$$= S_j = \sqrt{\frac{\sum_{i=1}^{N} (X_{ij} - \bar{X}_j)^2}{N - 1}} = \sqrt{\frac{N \sum_{i=1}^{N} X_{ij}^2 - (\sum_{i=1}^{N} X_{ij})^2}{N(N - 1)}} = \sqrt{C_{jj}}$$

Correlation (between variables j and k)

$$= R_{jk} = \frac{C_{jk}}{S_j S_k} = \frac{N \sum_{i=1}^{N} X_{ij} X_{ik} - (\sum_{i=1}^{N} X_{ij})(\sum_{i=1}^{N} X_{ik})}{\sqrt{N \sum_{i=1}^{N} X_{ij}^2 - (\sum_{i=1}^{N} X_{ij})^2}\sqrt{N \sum_{i=1}^{N} X_{ik}^2 - (\sum_{i=1}^{N} X_{ik})^2}}$$

From the equation $X_{ij} = B_{jk} X_{ik} + A_{jk}$, the program calculates

Linear Regression Coefficient (for predicting variable j from variable k)

$$= B_{jk} = R_{jk}\left(\frac{S_j}{S_k}\right)$$

Intercept (constant term in equation for predicting variable j from variable k)

$$= A_{jk} = \bar{X}_j - B_{jk}\bar{X}_k$$

*NOTE: the sample covariances and sample standard deviations are unbiased estimates of the corresponding population parameters. The definitions given here follow the practice of many current statisticians. [See Anderson (1958) - Chapter 3 for example.]

IV. Significance Tests

If we assume that two variables (indexed by j and k) have a bivariate normal distribution, there is a test statistic for testing the hypothesis that the correlation in the population is zero (or equivalently that either regression coefficient is zero). Even for a relatively small sample size (N). this hypothesis can be tested using the t ratio:

$$t = \frac{R_{jk}\sqrt{N - 2}}{\sqrt{1 - R_{jk}^2}}$$

with N - 2 degrees of freedom. Other types of hypotheses can be tested through use of the Fisher R to Z transformation. [See Hays (1963), pages 529 - 533 for example.]

V. Output

Output from the CORRELATION program may consist of any or all of the statistics from section III above, by using parameters 2 through 7. Any output from this program may be printed and/or output to temporary storage (SEQUENTIAL 1 - 5). The means, standard deviations, and the sample size (N) are output as a matrix with M rows (one for each variable) and three columns (the third column will have a constant value of N for all variables). Correlations, covariances, and cross-products are printed as lower triangular matrices, while the regression coefficients and intercepts are printed as square matrices. However, all five of these matrices are stored as square matrices.

VI. Restrictions

The CORRELATION program will accept an unlimited number of observations, but the number of variables is limited as noted in the section on PROGRAM LIMITS in the INTRODUCTION.

VII. Parameters

The parameters for the CORRELATION program follow the program name on the main program card. Each parameter must be enclosed in parentheses. The parameters must appear in the order given below. If a parameter is not needed, do not punch anything between its parentheses. All parentheses after the last non-empty pair may be omitted.

Parameter Number	Use or Meaning
1	Input Address of raw data (X matrix). CARDS or SEQUENTIAL 1-15 and/or PRINT.
2	Output Address for means, standard deviations, and sample size. SEQUENTIAL 1-15 and/or PRINT.
3	Ω Output Address for correlation matrix (R). SEQUENTIAL 1-15 and/or PRINT.
4	Ω Output Address for cross-products matrix. SEQUENTIAL 1-15 and/or PRINT.
5	Ω Output Address for covariance matrix (C). SEQUENTIAL 1-15 and/or PRINT.
6	Ω Output Address for matrix of regression coefficients (B). SEQUENTIAL 1-15 and/or PRINT.

7 Ω Output Address for intercepts (matrix A). SEQUENTIAL 1-5 and/or PRINT.

8 1 if last variable in each row is a weighting factor.

- It is possible to print in F format and/or punch the output from these parameters. If you need either of these options, see the section in the INTRODUCTION on INPUT and OUTPUT.

VIII. Special Comments

1. This program does not check for missing data. All blank spaces are read as zeroes. If you have missing data, use the MISSING DATA CORRELATION program.

2. In the output matrices of regression coefficients and intercepts, the row number refers to the dependent variables, and the column numbers refer to the independent variables.

3. If a variable is constant, an error message will be printed and all correlations with that variable will be set to zero.

4. In order to have the program perform its calculations separately for subsamples of the data, see the section on CONTROL VARIABLES in the INTRODUCTION.

IX. Examples

1A

```
/*ID <accounting information>
// EXEC SOUP
//SOUP.SYSIN DD *
CORRELATIONS (CARDS)( )(PRINT).
END SOUPAC
DATA (6)(6F2.0)
  .
  .
  .
END#
/*
```

1B

```
/*ID <accounting information>
// EXEC SOUP
//SYSIN DD *
COR (C)( )(P).
ENDS
DATA (6)(6F2.0)
  .
  .
  .
END#
/*
```

Example 1A illustrates the usage of the CORRELATION program. Notice that all words are spelled out although this is unnecessary. Notice also that correlations are to be printed out, although the means and standard deviations are not. Example 1B will perform exactly the same computations as 1A, except that all instructions have been abbreviated to make keypunching easier.

```
2
/*ID <accounting information>
//  EXEC  SOUP
//SYSIN  DD  *
COR (C)(P)(P/S1).
PRINCIPAL AXIS FROM (S1) TO (S2/P) WITH (10) FACTORS AND
(100) PERCENT OF THE VARIANCE TO BE REMOVED.
VARIMAX ROTATION FROM (S2) TO (PRINT).
ENDS
DATA (20)(10F4.0,5F6.2/10X,5F4.1)
   .
   .
   .
END#
/*
```

In the second example, the CORRELATION program first prints the means and standard deviations. Then it prints the CORRELATION matrix and stores it on SEQUENTIAL 1 (S1). The PRINCIPAL AXIS program then performs a principal components analysis and outputs 10 components to S2. VARIMAX then rotates these 10 components, using the VARIMAX criterion, and prints the results.

X. References

T. W. Anderson, An Introduction to Multivariate Statistical Analysis; John Wiley and Sons, Inc., 1958.

E. C. Bryant, Statistical Analysis; McGraw-Hill, 1960, pp. 113-135.

W. L. Hays, Statistics for Psychologists; Holt, Rinehart and Winston, 1960.

H. M. Walker and J. Lev, Statistical Inference; Henry Holt and Company New York, 1960.

MISSING DATA CORRELATION

I. General Description

The MISSING DATA CORRELATION program calculates the following coefficients for every combination of variables:

Mean: $X_i = \frac{\Sigma N_i}{N_i}$

Standard Deviation: $s_{xij} = [\frac{N_{ij}\Sigma(X^2{}_{ij}) - (\Sigma X_{ij})^2}{N_{ij}\ (N_{ij}-1)}]^{1/2}$

Covariance: $S_{ij} = \frac{N_{ij}\Sigma(XY_{ij}) - (\Sigma X_{ij})(\Sigma Y_{ij})}{N_{ij}\ (N_{ij}-1)}$

Correlation: $r_{ij} = \frac{S_{ij}}{Sx_{ij}Sy_{ij}}$

II. Restrictions

The maximum number of variables for this program is 100.

The input data to this program may come from any source conforming to SOUPAC. Output may be printed and the correlation matrix may be placed on any source conforming to SOUPAC.

III. Parameters

The parameters for the MISSING DATA CORRELATION program appear on the program card. They must follow the program name in the following order:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15. Default is CARDS.
2	0 - printing as usual 1 - printing is suppressed
3	Output Address of correlation matrix.
4	Output Address for sample sizes.
5	Coding for missing data; if left blank or if zero is entered, minus zero is used as check. It is NOT possible for this program to count true zeroes as missing data. This parameter must be enclosed in asterisks. Example: *99*.

NOTE: All output is in double precision.

[illegible]. Special Comments

A. The user is warned against further processing of the correlations output by this program because the correlations do not necessarily come from the same sample.

B. For control breaks, data must be presorted on the control variables with the last variable changing fastest. The maximum number of control variables is 30 Control variables begin on a new card with ‡C-B in column 1 and are enclosed in parentheses.

C. The correlation matrices can be stored in parameter 3 is a temporary storage address. However, if control breaks are also being used, only the first matrix corresponding to the first control break can be saved.

PARTIAL CORRELATION

I. General Description

This routine, upon option, provides two of the more common types of special purpose correlation coefficients.

A. Partial Correlations:

This program produces coefficients of net correlation of any order from 1 to 19 in matrix form. Coefficients of successively higher order may be obtained by repeated calls to the program, each time using as input the previously generated partial correlation matrix; or several variables may be held constant at the same time by one call to the program.

The general equation used is:

$$r_{ij.abc...n} = \frac{r_{ij.abc...(n-1)} - r_{in.abd...(n-1)} \cdot r_{ij.abc...(n-1)}}{(1 - r^2_{in.abd...(n-1)})^{1/2} \, (1 - r_{jn.abc...(n-1)})^{1/2}}$$

References:

Mills, F.C. Statistical Methods, Holt, Rinehart and Winston. New York, 1955, 3rd edition.

B. Tetrachoric Correlations:

This type of correlation coefficient is used when continuous normally distributed variables are measured dichotomously.

This program is based on a program by Roald Buhler at Princeton University which in turn is based on a 650 program written at the Educational Testing Service. The approximation used was developed by Professor Ledyard Tucker.

II. Restrictions

A. Partial Correlations:

Input matrices may be no larger than 140 x 140 and must be compatible with SOUPAC conventions. In most cases the original input to the program will be a matrix of zero order correlations (see CORRELATION program write-up).

B. Tetrachoric Correlations:

This option is limited to 140 variables. All observations should be coded either 0 or 1. The program generates cross-count tables before computing the correlation coefficients.

III. Parameters

The program name, PARTIAL CORRELATION, should be followed by the following parameters:

Parameter Number	Use or Meaning
1	Input address of R if partial correlations or raw data if tetrachoric correlations. (R is a correlation matrix).
2	Output Address of correlations desired.
3	0 if tetrachoric correlations are desired 1 if partial correlations are desired
4 - 22	Variables to be held constant in using partial correlations.

IV. Special Comments

When there is a zero cell or sufficiently close so that the tetrachoric correlation cannot be computed by this approximation, a value of -1.0 is used if the missing cell is off-diagonal. If a diagonal cell is zeroish (i.e., if a variable is all zero or all one) its correlations are set to 0.0.

Blanks are counted as zeroes.

V. Examples

A series of observations of 8 variables are used to obtain 3rd order partial correlations with variables 5, 7, and 8 held constant:

```
/*ID
// EXEC  SOUPAC
//SYSIN  DD  *
CORRELATIONS (CARDS)()(SEQ 1).
PARTIAL  CORRELATION (SEQ 1) (PRINT)(1)(5)(7)(8).
END SOUPAC
DATA(8)(8F6.2)
    .
    .
    .
    .
END #
/*
```

STEP-WISE MULTIPLE CORRELATION

I. General Description

The STEP-WISE MULTIPLE CORRELATION program calculates the following:

Mean: $\bar{X}_i = \frac{\Sigma X_i}{N}$

Covariance: $s_{ij} = \frac{N\Sigma(X_i X_j) - (\Sigma X_i)(\Sigma X_j)}{N(N-1)}$

Standard Deviation: $s_i = (s_{ii})^{1/2}$

Product Moment Correlation: $r_{ij} = \frac{s_{ij}}{s_i s_j}$

In the step-wise procedure, intermediate results are used to give valuable statistical information at each step in the calculation. These intermediate answers are also used to control the method of calculation. A number of intermediate regression equations are obtained by adding one variable at a time thus giving the following intermediate equations.

a. $Y = B_0 + B_1 X_1$

b. $Y = B_0 + B_1 X_1 + B_2 X_2$, etc.

The coefficients for each of these intermediate equations and the reliability of each coefficient are obtained by the step-wise procedure. The values and reliability may vary with each subsequent equation. The coefficients represent the best values when the equation is fitted by the variables included in the equation. The variable is added that makes the greatest improvement in "goodness of fit" or, stated another way, gives the greatest reduction in variance of the dependent variable.

A variable may be indicated to be significant at an early stage and enter the regression equation. After several other variables are added to the regression equation, a variable in the equation may be indicated to be insignificant. Under this situation the step-wise regression procedure will remove the insignificant variable before adding an additional variable. Thus, at the various steps in the regression procedure, only those variables which are significant will be included in the regression equation.

The F level to enter a variable controls when variables enter the equation and the F level to remove a variable likewise controls the removing of variables from the equation.

The last step in the step-wise procedure predicts the value of the dependent variable for each set of observations based on the final regression equation. Deviation between the actual and predicted values are also calculated.

(See parameter 4).

For reference to formulas and methods used see:

> A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, New York, Wiley and Sons, 1960, pp. 191-195.

II. Restrictions

The maximum number of independent variables in this program is 199. The dependent variable must be the last variable of each row.

The input data to this program may come from any source conforming to SOUPAC. Output may be PRINT only.

III. Parameters

The parameters for the STEP-WISE MULTIPLE CORRELATION program appear on the program call card. They must follow the program name in this order:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15. (See parameter 4 for special conditions).
2	"F" level to enter an independent variable into the regression equation. An example would be: *4.0*
3	"F" level to remove a variable from the regression equation. An example would be: *4.0 *
4	This parameter should be set to 1 if the predicted dependent variables are to be calculated. (If this option is needed, input data must not be from cards.) 0 or blank if not wanted.
5	1 if constant term in equation is assumed to equal zero (0).
6	1 if want to use weighting factor. (If a weighting factor is used, it must be the last variable in the input data row.)
7	1 if intermediate steps of regression are not to be printed.
8	1 if do not want cross-product matrix printed 2 if input data is in the following form:

	1 N	N+1
N	CORRELATION MATRIX	MEANS
N+1	STANDARD DEVIATION	sample size

(where N is the number of variables)

Parameter Number	Use or Meaning
9	1 if do not want means and standard deviation printed
10	1 if do not want covariance to be printed
11	1 if do not want correlations to be printed.
12	Tolerance to be used to determine when singularities are assumed to occur. If this parameter is left blank 10^{-5} is used. If it is desired to change this parameter, the following would be used: *1.E -10* where any number could be substituted for the 10.
13	Output (intermediate storage) of coefficients.
14	First (N) variables are placed in regression first.

IV. Special Comments

The dependent variable must be the last variable in the input row (unless a weighting factor is used, then the dependent variable will be the next to the last variable in the input row).

Negative F-ratios may sometimes result in the computational procedure. They should be considered to be analytically zero. Frequently negative F's arise when input is a missing data correlation matrix. Results up to the negative F are always correct.

The standard error of Y given at each step is the standard error of predicted Y.

The program will loop if a variable entered at one step is removed at the very next step. This can usually be corrected by changing the F-levels to enter and remove variables.

V. Example

```
STEP(CARDS)*4.**4.*( )( )( )( )(1).
```

DISTRIBUTION ANALYSIS PACKAGE

FIT
(CHI-SQUARE GOODNESS-OF-FIT TEST)

I. General Description

In statistical applications, it is frequently the case that certain assumptions were made concerning the probability distribution of a random variable. A frequent assumption is that a particular variable is normally distributed. The question that arises is "how valid an assumption was this?" A method of testing this assumption (or hypothesis) is the CHI-SQUARE GOODNESS-OF-FIT TEST.

This program provides tests of hypothesis that user's data is (1) a random sample from the distribution $P\Theta$, where Θ is a user specified parameter, or (2) a random sample from a class, $P\Theta$, of distributions where Θ is not specified. These will be called tests of type 1 and type 2 , respectively.

Distributions which can currently be tested in the program are the binomial, Poisson, normal, gamma and continuous rectangular distributions. The user may, for example, wish to test the hypothesis that his data was from a normal distribution with variance 1, for some mean.

The program on the basis of the user provided information decides on a set of points $X_1 < X_2 < \ldots . X_n$ in the range of the distribution under consideration. The test then compares the observed numbers, o_i ,of observations in each interval $[X_{i-1}, X_i)$, with the expected numbers

$$e_i = P_{\hat{\Theta}} \{X_{i-1} \leq Y_1 \leq X_i\} \quad , \quad \{Y_i \; ; 1 \leq i \leq m \}$$

Where $\hat{\Theta}$ is the user specified value of Θ , if it was specified, and if not, $\hat{\Theta}$ is the maximum likelihood estimator for a hypothesis of form (2) above. The comparison of o_i with e_i is made by the statistic

$$X^2 = \sum_{i=1}^{n+1} \frac{(o_i - e_i)^2}{e_i} \quad .$$

The distribution of this statistic has an approximate X^2 distribution when the hypothesis is true and n is large. (See Billingsley, 1961). The program prints the computed value of X^2 and the number of degrees of freedom for the test.

If one sample, with m observations, is to be tested, input to the program from cards or sequential will be one variable with m observations. More than one sample may be tested at a time, if the same distribution, or in some cases a distribution from the same class, is the one being considered. Thus a test of each of three samples, all of which are on the same sequential storage or card deck, may be performed.

II. Distributions Available

NOTATION: In the description below f(x) denotes the value of the probability density function of the point x. F(x) is the cumulative distribution function defined as $\Pr\{X \leq x\}$

A. DISCREET DISTRIBUTIONS

The Distributions are classified here as being "discreet," (i.e. having positive probability at a countable number of values of the random variable it describes) or as "Continuous" (having positive density over a continuous range of values of the random variable).

1. BINOMIAL(N,P) N a positive integer; $0 \leq p \leq 1$

$$f(x) = \binom{N}{x} p^x (1-p)^{N-x} \qquad \text{for x an integer such that } 0 \leq x \leq N$$

$$F(x) = \sum_{i=0}^{x} \binom{N}{i} p^i (1-p)^{N-i} \qquad \text{for x an integer such that } 0 \leq x \leq N$$

This distribution is appropriate, for example, in situations where a random variable is sampled independently (observed) N times, the observations, and where the values of the random variable can be classified precisely as being in one of two sets often denoted "success," and "failure," with probabilities p, and q = 1 - p respectively.

2. POISSON (λ) $\lambda > 0$

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!} \qquad x \geq 0 \text{ ; and x an integer}$$

$$F(x) = \sum_{i=0}^{x} \frac{\lambda^i e^{-\lambda}}{i!} \qquad x \geq 0$$

The Poisson distribution is often used as an approximation to the binomial when the number of observations is large, p = Pr {success} is small, and the product n * p is essentially a constant. The Poisson is a pervasive distribution in its own right, arising in situations where the probability of no occurrences, or of one occurrence of a phenomenon in a unit of time is moderate, where the probability of more than one occurrence is essentially negligible in comparison, and where the frequencies of occurrence in adjacent intervals are independent of each other.

B. CONTINUOUS DISTRIBUTIONS

1. NORMAL (μ, σ^2) $-\infty < \mu < +\infty$; $\sigma^2 > 0$

$$f(x) = \frac{1}{\sqrt{2\Pi\ \sigma^2}} \exp \frac{-(X - \mu^2)}{2\ \sigma^2} \qquad -\infty < x < +\infty$$

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\ \Pi\ \sigma^2}} \exp \frac{-(t - \mu)^2}{2\sigma^2}\, dt \qquad -\infty < x < +\infty$$

Through the Central Limit Theorem, the use of this distribution has been justified to describe a tremendous variety of phenomena in which the random variable under consideration is assumed to be the sum of a large number of independent random variables, each having a small contribution to the total.

2. GAMMA (α,β) $\alpha > 0$, $\beta > 0$

$$f(x) = \frac{\beta^{\alpha}\ x^{\alpha-1}\ e^{-\beta x}}{\Lambda(\alpha\ )} \qquad \text{for } x > 0$$

$$F(x) = \int_{0}^{x} \frac{\beta^{\alpha}\ t^{\alpha-1}\ e^{-\beta t}\ dt}{\Lambda(\alpha)}$$

Here $\Lambda(\alpha) = (\alpha-1)\,\Lambda(\alpha-1)$ and so if α is a non-negative integer, $\Lambda(\alpha) = (\alpha-1)!$

The gamma distribution is the sampling distribution for the sum of α independent identically distributed "negative exponential" random variables, with $\beta = \lambda$ where λ is the parameter for the negative exponential distribution. The negative exponential distribution is itself the special case gamma $(1, \beta)$. The gamma distribution is used in dealing with waiting times, where the expected frequency in a given interval has Poisson distribution. The "Chi-Square" distribution is another important special case of the gamma distribution where

$$\chi^2\ (r) \equiv \text{gamma}\ (r/2,\ 1/2).$$

3. RECTANGULAR (Θ_1, Θ_2) $\Theta_1 < \Theta_2$

$$f(x) = \frac{1}{(\Theta_2 - \Theta_1)} \qquad \Theta_1 \leq x \leq \Theta_2$$

$$F(x) = \int_{\Theta_1}^{x} \frac{1}{\Theta_2 - \Theta_1}\, dt = \frac{x-\Theta_1}{\Theta_2-\Theta_1} \qquad \Theta_1 \leq x \leq \Theta_2$$

This distribution is appropriate where any event in an interval $\{\Theta_1, \Theta_2\}$ has equal probability of occurring, but no occurrences will be outside the interval.

III. Use of the Program

A. PRELIMINARY

The program is invoked by the main parameter card (a SOUPAC "Program Card"). Information concerning the distribution and intervals to be used are supplied on subparameter cards for the program.

Input is in the form of column vectors, and may be from cards or sequential storage. Each variable (vector of data will be tested against the same type of distribution, i.e. information provided on the distribution card applies to all variables.

The subparameter card provides the means of supplying information to the program concerning the distribution to be used, as well as the number and size of intervals to be used in the tests. In some cases a distribution parameter must be supplied by the user.

B. MAIN PARAMETER CARD

Immediately following the program name CHI-SQUARE GOODNESS-OF-FIT (mnemonic: FIT), the following parameters are listed.

Parameter Number	Use of Meaning
1	Input Address. May be CARDS or SEQUENTIAL (S1, or S2, etc).

C. SUBPARAMETER CARDS

1. GENERAL DESCRIPTION

One "distribution card", chosen from the list below, should appear with each call to the program. For tests of type 1 ("simple hypothesis") the parameters of the distribution are specified on this card. For distributions allowing tests of type 2 ("composite hypothesis") one or more of these parameters may be left blank. Other parameters on the card relate to intervals to be used for determining observed and expected frequencies. Because this test is "asymptotically valid," these should be chosen so that expected frequencies of any interval does not fall below 5.

2. DISTRIBUTION CARDS

Where * * is used, a parameter requires a decimal point number; when ( ) appears an integer is required.

DISTRIBUTION

BINOMIAL(M)*P*(N).

M and P are parameters of the distribution, P the probability of success on a trial, M the number of trials. N is the number of points (integral values starting at 0) to be grouped in each interval. For a test of Type 2, the parameter P should be left blank.

POISSON*λ*(ENDPOINT)(N).

λ is the Poisson density parameter, and may be left blank for a test of type 2. N is the number of adjacent points to be grouped in each interval until ENDPOINT is reached. The Interval [ENDPOINT, $+\infty$) is then the last interval.

RECTANGULAR*Θ1**Θ2*(N).

Θ1, and Θ2 are the endpoints for the range of the distribution $\Theta 1 < \Theta 2$. N is the number of equal sized intervals to use for the test. Only tests of type 1 are allowed with this distribution.

NORMAL*μ**σ^2**STPT**ΣP*(N).

μ is the mean of the distribution, σ^2 is the variance. Either or both of the parameters may be left blank for a test of type 2. STPT and ΣP are the start point and end point of an interval to be broken up into N equal sized intervals for the test. The additional intervals ($-\infty$, STPT), and [ΣP, $+\infty$) will be used.

GAMMA*α**β**ENDPOINT*(N).

α is the degree's of freedom parameter, and must be specified. *β* is the density parameter and may either be specified for a type 1 test or left out for a type 2 test. The portion of the real line between 0 and ENDPOINT will be divided into N equal sized intervals for the test. One additional interval, [ENDPT, $+\infty$) will be included.

IV. Restrictions

A maximum of 50 input variables (each variable a sample to be tested) may be input to the program. Only one distribution or distribution type (e.g. NORMAL with variance 1 and any mean) may be tested.

V. Examples

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
FIT(CARDS).
NOR*2.0****-10**14*(24).
END P
END S
DATA(1000,1)(21X,F4.1)
.
.
.
.
#END
/*
```

This program will test the hypothesis that the sample is from a normal distribution with a mean of 2, for some variance. The interval [-10,14) will be divided into 24 pieces, each of length 1, for the test.

```
/*ID
//  EXEC  SOUP
//SYSIN  DD  *
FIT(CARDS).
BIN(10)*1/2*(2).
END P
END S
DATA(600,1)(26X,F2.0)
.
.
.
.
#END
/*
```

This program will test the simple (type 1) hypothesis that the sample was from a B (10,1/2) population. Note that with the given number (600) of observations we had to group adjacent pairs of points into the same interval to insure a reasonable expected frequency in every cell.

THE KOLMOGOROV-SMIRNOV STATISTIC

I. General Description

The program computes the Kolmogorov-Smirnov (K-S) D statistic,

$$D = \sup_{\text{all } x} \left| F_N(x) - F(x) \right| ,$$

where F_N is the sample cumulative distribution for a sample of size N and F(x) is the specified cumulative distribution.

II. Theoretical Discussion

The sample distribution function:

$$F_N(x) = \frac{j}{N} \qquad j = 0, \ldots\ldots\ldots N$$

will generally differ from the population distribution function. If the sample distribution differs exceedingly from the specified distribution F(x), the amount of the difference might be of use in determining whether to accept the hypothesized distribution as correct. The Kolmogorov-Smirnov test uses the maximum actual numerical difference $\left| F_N(x) - F(x) \right|$.

Example:

Consider testing the hypothesis that a distribution is normal with mean = 32 and variance=3.24 with 10 sample observations

31.0	33.7
31.4	34.4
33.3	34.9
33.4	36.2
33.5	37.0

$F_N(x)$ and F(x) are sketched below.

D = .56 which is the maximum of $\left| F_N(x) - F(x) \right|$.

At the .95 confidence level the critical value D' = .40925

Since D > D' the distribution being tested is rejected at the 5% level.

* See Owen [5]

III. Notes

Only supply the parameters needed to determine a specified distribution. This program, at this time, calculates Kolmogorov-Smirnov statistic for the following distributions if given the proper parameters.

A. Normal
   1. mean
   2. variance

B. Chi-Square
   1. Degrees of freedom

C. Central F
   1. Degrees of freedom numerator
   2. Degrees of freedom denominator

D. Noncentral F
   1. Degrees of freedom numerator
   2. Degrees of freedom denominator
   3. noncentrality parameter
      when noncentrality parameter $\lambda = \frac{1}{2} \sum_{i=1}^{N} \mu^2$

(See Graybill [3] for further development of λ).

Additional distributions are planned for the near future. Please see consultants for these additions.

IV. Sample Programs

A. Normal - type 1
   = 32
   $\sigma^2 = 3.24$

```
K-S(C)(1)(32***3.24).
```

B. Chi-Square - type 2
   d.f.= 24

```
K-S(C)(2)****(24).
```

C. Central F - type 3
   d.f. numerator = 5
   d.f. denominator = 16

```
K-S(S1)(3)****(5)(16).
```

D. Noncentral F - type 3
d.f. numerator = 10
d.f. denominator = 13
noncentrality parameter (λ) = 2.5

```
K-S(S2)(3)****(10)(13)*2.5*.
```

V. Parameters

The following parameters follow the mnemonic K-S:

Parameter Number	Use or Meaning
1	Input address, CARDS, SEQUENTIAL 1-15.
2	Specified distribution (see Section IV).
3	Floating point value of the mean.
4	Floating point value of the variance.
5	Degrees of Freedom (numerator).
6	Degrees of Freedom (denominator).
7	Floating point value of non-centrality parameter.

VI. References

[1] Darling, D. A., "The Kolmogorov-Smirnov, Cramer-Von Mises Tests," Annals of Mathematical Statistics, Vol. 28, (1957), pp. 823-838.

[2] Goodnight, James, Department of Experimental Statistics, N. C. State University, Raleigh , North Carolina. Mr. Goodnight developed and programmed the computational procedure used to integrate the central and non-central F distribution.

[3] Graybill, F. A., An Introduction to Linear Statistical Models, Vol. I, Chapter 4, "Distribution of Quadratic forms" pp. 74-92.

[4] Lindgren,

[5] Owen, D. B., Handbook of Statistical Tables. This book contains a table of the critical values for the Kolmogorov-Smirnov statistic. It is available for reference in the SOUPAC office.

FACTOR ANALYSIS PACKAGE

Factor Analysis Package

Users of the PRINCIPAL AXIS FACTOR ANALYSIS program are reminded that if no estimates of communalities are provided with the input matrix, the output is a principal components matrix. If communalities are provided, then the output is principal axis factors. Whether communality estimates are used, and if so, what kind, can be a very important conceptual consideration. Mathematically, the principal components approach the principal axis factors for large numbers of variables. The rate of convergence is the ratio of N^2 to N^3.

Centroid factor analysis similarly requires that a user consider the question of communality estimates.

The ITERATIVE FACTOR ANALYSIS program on the other hand, simultaneously estimates communalities and factor loadings.

The UNRESTRICTED MAXIMUM LIKELIHOOD procedure requires as input a correlation matrix with unities on the diagonal.

For principal components, component scores are derived by the equation $ZR^{-1}F = X$ where Z is the standardized raw scores; R^{-1} the inverse of the correlation matrix; and F the component loadings.

In general, factor scores from oblique rotations are obtained by the formula $X = DY^{-1} M^{-1} F'Z$ where D is the correlation between components and the reference vectors; Y the reference vector transformation; M the roots; F the principal components; and Z the standardized raw scores. For the orthogonal case, the general formula is $X = TM^{-1} F'Z$ where T is a transformation matrix, for example, the varimax loadings. Notice that if T is the identity matrix we have the formula for component scores.

These general formulas are also applicable to principal axis factors, as well.

BINORMAMIN

I. General Description

BINORMAMIN ROTATION rotates a matrix, F, of orthogonal factor loadings to oblique simple structure.

It does this by iterating for T in FT = A (where A is the rotated factor pattern) so as to minimize:

$$K = \sum_{p} K_{p} = \sum_{p} \quad \overset{q}{q=p} \left[\frac{\sum_{j} (v_{jp}^{2}/h_{j}^{2})(v_{jq}^{2}/h_{j}^{2})}{(\sum_{j} v_{jp}^{2}/h_{j}^{2})(\sum_{j} v_{jq}^{2}/h_{j}^{2})} \right]$$

Since solving directly for K is too complex, BINORMAMIN takes one vector at a time, rotating it against all the others, to minimize each K_p.

Its name comes from the fact that is uses a double (BI) NORMAlization in seeking a MINimum.

For further information see:

1. Kaiser, H. F. and Dickman, K. W., "Analytic Determination of Common Factors". Unpublished manuscript, University of Illinois, 1959.

2. Harmon, H. H., Modern Factor Analysis . Chicago, University of Chicago Press, 1960. pp.326ff.

II. Restrictions

Input is limited to matrices of 150 x 30 or less.

III. Parameters

After the program name, BINORMAMIN, are the following parameters:

Parameter Number	Use or Meaning
1	Input Address of factor matrix, F.
2	Output Address of factor matrix, F.
3	Output Address of the transformation matrix, T.
4	Output Address of the reference vector structure, V.
5	Output Address of the correlations between reference vectors.
6	Output Address of the primary factor pattern, P.

Parameter Number	Use or Meaning
7	Output Address of the correlations between factors.
8	Maximum number of iterations (see note). If blank, the maximum will be set at 100 iterations.
9	Convergence criterion (see note). A. Defined zero change: iterating will stop when each element in V changes by less than A. (A must be less than .2 and no less than .0000001). B. Defined zero rotation: iterating will stop when each vector in T changes by less than θ, where θ is the angle whose cosine is B. (B must be less than 1.0 and no less than .2). If left blank, A will be set to .001.
10	If an initial T is to be read in, input address of T.
11	Output Address of the initial T.

Note on Output: A. Any output option left blank will not be output.

B. The program will always print out the program name and the number of iterations actually done.

C. The program will print out the largest change in V, unless option 9B is used.

D. All data printed out is to 7 decimal places.

Note on parameters 8 and 9: Program will stop at whichever criterion it meets first.

Note on parameter 9: This parameter is a floating point constant and therefore must be enclosed in asterisks, with a decimal point, as in example:

```
Example: BINORMAMIN (CARDS)()()(SEQ 1/PRINT)()()(PRINT)()*.0001*.
```

Store V on SEQ 1, also prints V and correlations between factors. On the last iteration, no element in V changed by more than .0001, unless the maximum of 100 iterations was reached.

CENTROID FACTOR ANALYSIS

I. General Description

CENTROID FACTOR ANALYSIS computes a set of f linearly independent vectors (factors) which are mutually uncorrelated. Normally, a factor analysis decomposes a matrix of correlations, R_n, into a set of f factors. The factors are arrayed as column vectors in the factor matrix, F, such that

$$R_n = FF' + R_{(n-f)}$$

where R_{n-f} is the matrix of residual effects. The K^{th} factor is computed by dividing the column sums of R_{n-k} by the square root of the total sum of elements of R_{n-k}

$$fi,k = \Sigma\, r_{i,j}^{(h)} \Big/ \sqrt{\sum_{ij}\sum r_{i,j}^{(k)}}$$

Between each factor extraction, the variables in the residual matrix are successively reflected until all the column sums are positive.

For more detailed discussion see:

1. L. L. Thurstone, Multiple Factor Analysis, Chicago, University of Chicago Press, 1947, pp. 149-175.

2. Harry Harmon, Modern Factor Analysis, Chicago, University of Chicago Press, 1960, pp. 192-215.

II. Restrictions

The input matrix for the CENTROID program must not exceed the dimensions of 190 x 190. The input matrix is further limited to being a square, positive definite or semi-definite, symmetric matrix. Commonly, correlation, covariance, or cross-product matrices are used as input data. Any attempt to introduce communality estimates (change the diagonal elements) must be made before data is passed to the CENTROID program. A set of communalities, which are incorrectly estimated, can make the matrix non-positive and could conceivably cause a hang-up.

The input data may come from any storage medium which conforms to SOUPAC. Similarly, the output codes follow the established conventions and are at the option of the user.

The input matrix may be completely factored (i.e., N factors from a N variable matrix). However, factoring may be stopped by any of three criteria:

1. The user may specify the number of factors to be extracted. This criterion provides an upper limit beyond which factoring will not be done. Consequently, it is advisable to put the maximum value on this limit in cases where it is not the primary criterion. (Set it equal to the number of variables).

2. The per cent of total variance removed from the R matrix is a second limiting criterion. This parameter also specifies an upper limit to the process. Therefore, it should be set at 100 per cent unless it is the criterion for stopping.

3. The last criterion is to stop when the factor contribution falls below 1. The use of this procedure is dictated by the presence or absence of its associated parameter.

If all three criteria are used simultaneoulsy, factoring will be stopped by whatever criterion is met first.

III. Parameters

Following the program anme on the program call card come the parameters needed by the program. The parameters must appear in the order below:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15.
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.
3	Maximum number of factors to be extracted. This must be less than or equal to the order of the input matrix.
4	Per cent of total variance to be removed expressed as an integer between 0 and 100.
5	The presence of any number greater than 0 in this parameter indicates that factoring should stop when the factor contribution falls below unity.
6	Output Address of Residual Matrix.

If parameters 3 and 4 are left blank then by default option they will be set to maximum possible values and a message will be printed.

Residual Matrix <u>must</u> be stored before it can be printed.

Example: Assume that you have 77 variables and that the correlation matrix is stored on SEQ 1, then legal forms of CENTROID call statement may be:

```
CENTROID(SEQ 1)(PRINT)(77)(100)(1).
CENTROID  (SEQ 1)(P(F))(50)(80).
CENTROID (SEQ 1)(SEQ 2/P)(20)(100)(1)(SEQ 3/P).
CENTROID (SEQ 1)(SEQ 2/P(F))(15)(90)()(SEQ 3/P(F)).
CENTROID (SEQ 1)(P).    In this case, number of factors = 77
and per cent of variance = 100 will be assumed by default.
```

COMMUNALITY ESTIMATION

I. General Description

Five methods of COMMUNALITY ESTIMATION are offered in this program. In each case the estimates replace the diagonal elements of the matrix. They are as follows:

Code Number — **Method**

1 — The element of largest absolute magnitude in each row replaces the diagonal element of the row.

2 — The square of the multiple R of each variable with all others replaces the diagonal entry for that variable. (See Special Comment Number 2).

3 — Communalities produced from another analysis and are to be input from cards or another storage medium.

4 — For each row (N) $((\sum_{j=1}^{N} r_{i,j}^{2})/N)^{1/2}$ replace the diagonal entry for that row. This is the square root of the average square across the row.

5 — For each row (N) $(r^*_{ik})(S_i - r^*_{ik})/(S_k - r^*_{ik}$ replaces the diagonal entry for that row where:

$$r^*_{ik} = \max \text{ abs } (r_{ij}) \text{ and}$$

$$S_i = \sum_j \text{abs } (r_{ij}),\quad S_k = \sum_j \text{abs } (r_{kj})$$

This method of COMMUNALITY ESTIMATION is due to Professor L. Tucker.

II. Restrictions

Input is restricted to correlation matrices of order 150 or less.

III. Parameters

The parameters for the COMMUNALITY ESTIMATION program appear on the program call card. They must follow the program name in this order:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15.

Parameter Number	Use or Meaning
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.
3	Section Code Number. (See General Description).
4	Input Address if Option 3 is used.

IV. Special Comments

(1) If the correlation matrix and communalities both are input from cards, the correlation matrix precedes the communality estimations. (See Code Number 3 in the General Description).

(2) If the input correlation matrix is singular, or very nearly so, squared multiple correlations computed by standard procedures may be subject to considerable error, and will usually exceed unity for several variables. For this reason, the user should be aware of the characteristics of the matrix. The program will check to see that all R^2 are less than or equal to 1.0. If this is not the case, execution will cease and a message will be printed stating that the correlation matrix is virtually singular. There is an alternative procedure devised by Ledyard Tucker for finding R^2 in singular matrices. Information is available in the SOUPAC office.

V. Reference

Harmon, Harry: Modern Factor Analysis, Chicago, University of Chicago Press, 1960. pp. 83-90.

ITERATIVE FACTOR ANALYSIS

I. General Description

A. Procedural

This routine, upon option, provides one of four iterative factorization methods:

1. Alpha factor analysis (AFA, Kaiser, 1962)
2. Canonical factor analysis (CFA, Rao, 1955, Harris, 1962)
3. Stepwise maximum likelihood factor analysis (MLFA, Lawley, 1940)
4. Iterative principal axis factor solution (IPRAX, Traditional)

All four methods have in common that communalities and factor loadings are estimated simultaneously. In three cases (AFA, CFA, IPRAX) the number of factors decision can be made beforehand by the user, or it can be left to the program, in which case appropriate modifications of Guttman's lower bound criterion will be used.

The four methods differ from each other in theory with respect to the defining criterion of optimization, and consequently they differ technically with respect to the matrix that is diagonalized in each case.

1. AFA (Kaiser)

Optimization criterion: maximize the alpha-reliabilities (Cronbach) of the retained factors. If the number of factors decision is left to the program, the Kaiser modification of the Guttman criterion will be used and all factors with positive alpha-reliability will be iterated upon.

The diagonalization is on the matrix C in

$$C = H^{-1} (R - U^2) H^{-1} \quad \text{so that } C = Q\theta Q'$$

where R is an nxn input matrix of covariances, H^2 is a diagonal matrix of communalities. $U^2 = I - H^2$ is a diagonal matrix of uniquenesses, Q is an nxm matrix of latent vectors corresponding to the m largest latent roots in θ which are used to recompute new estimates of H^2 through $F = H\, Q\, \theta^{1/2}$. An initial set of H^2 is provided by $I - (\text{diag}(R^{-1}))^{-1}$ which is equivalent to the squared multiple correlations of R if R itself is a correlation matrix.

Invariance under scaling: Kaiser has shown that the resulting factors will be invariant under scaling, i.e., if a covariance matrix R gives rise to a factor matrix F then the covariance matrix SRS will give rise to a factor matrix SF (S diagonal).

Behavior of latent roots, alpah reliabilities: the n-m rejected roots of C add to zero at each state (i.e., C is non-Gramian), the m accepted roots are simple functions of the alpha-reliabilities of the retained factors in F. These reliabilities will be output by this sub-program.

2. CFA (Rao, Harris)

Optimization criterion: maximize the correlations between m linear combination of the common parts of the variables with m linear factors that are canonically correlated (Hotelling) with the variables in the common factor space. If the number of factors decision is left to the program, the Harris modification of the Guttman criterion will be used, leading to a Gramian $R-U^2$ of minimum rank.

The diagonalization is on the matrix

$$C = U^{-1} (R - U^2) U^{-1} \quad \text{so that } C = Q\theta Q'$$

where $F - UQ\theta^{1/2}$ is used to recompute new estimates for U^2, retaining the m largest roots of C in θ. The notation is the same as in section 1 (AFA). An initial set of U^2 is provided by $[\text{diag } (R^{-1})]^{-1}$.

Invariance under scaling: the resulting factors are again invariant under scaling as defined in section 1 (AFA).

Behavior of latent roots: Chi-square criterion: Rao has shown that the n-m rejected roots approach unit at convergence. For exact rank m data they will be "exactly" unity within the tolerance of the convergence criterion ETA (see section III - B). For data containing random error their departure from unity provides a likelihood ratio test for the hypothesis that the population matrix $P-V^2 = GG'$, where P, V, G are population parameters corresponding to R, U, F in the sample, is rank m or less. A criterion for this test is computed by this sub-program which can be compared with two chi-square approximations which are also output by this sub-program. Note that such a chi-square test is valid only if the iterative process has indeed converged, as indicated by the maximal discrepancy between trial vectors which is printed out for that purpose.

3. MLFA (Lawley)

The CFA variant of the program can be used for a step-wise maximum likelihood factorization in the Lawley-Rao sense.

Optimization criterion: maximize the likelihood function corresponding to the multivariate normal distribution with covariance matrix parameters $P = GG' + V^2$ (as defined in section 2, CFA), given the sample matrix R, under choice of G and $\underline{V}$ and observing the side-conditions that $P - V^2$ is Gramian and V^2 diagonal with $0 < v_i < 1$.

The diagonalization is the same as in CFA, section 2, hence, the resulting factors are again invariant under scaling as defined in section 1 (AFA).

In contrast to CFA, however, the number of factors decision is made on statistical ground. The user would start with a reasonable guess for m (preferably $m < n/2$ to ensure positive degrees of freedom for the chi-square test, which otherwise will be bypassed). After convergence has been obtained, the user would insepct the chi-squared statistic. If the statistic is below the table values of the chosen porbability level (.05 or .01), then the hypothesis can be accepted at this level with corresponding risk and the user has the option to reduce m for a second run, etc. On the other hand, if the adjusted statistic exceeds the table value, then m must be raised until the adjusted statistic warrants acceptance of the hypotheses.

Within the package the user is free to re-enter the routine repeatedly with sequentially de- or increasing m specified on the call card. Since the test assumes convergence, it is pertinent that the number of iterations be allowed large enough for convergence to occur within the chosen tolerance bound ETA (see section III - B).

4. IPRAX (traditional)

<u>Optimization criterion</u>: none

The diagonalization is on the matirx

$$C = R - U^2 \quad \text{so that} \quad C = Q\theta Q'$$

where $F = Q\theta^{1/2}$ is used to recompute $H^2 = I - U^2$, retaining the m largest roots in θ. The notation is the same as in section 1 (AFA). An initial set of H^2 is provided by the identity matrix. If the number of factors decision is left to the program, the unmodified Guttman criterion will be used, i.e., all factors corresponding to roots of the input matrix R which exceed unity will be retained.

<u>Invariance under scaling</u>: as defined in section 1 (AFA) is not obtained by this method.

The behavior of the latent roots is not known at present. No statistical or other significance can be attached to the m largest or n-m smallest root of C.

5. Both covariance matrices and correlations matrices are acceptable as input. If covariances are used the tenth parameter should be 1. In this case the covariance matrix is scaled into a correlation matrix, and all computations, in particular the number of factors decision, are based on this correlation matrix. At the final stage the factors are scaled back so as to account for the covariance matrix which was input. The matrix of residuals is computed in the metric of the covariances.

References:

Guttman, L. "General Theory and Methods for Matrix Factoring," Psychometrika, 1955, IX, 1-16.

Harris, C. L. "Some Rao-Guttman Relationships," Psychometrika, 1962, XXVII, 247-263.

Hotelling, H. "The Relation of The More Multivariate Statistical Methods to Factor Analysis," British Journal of Statistical Psychology, 1952, X, 69-79.

Kaiser, H. F. and Caffres, T. Psychometric Method of Factor Analysis. Unpublished manuscript, 1963.

Lawley, D. N. "The Estimation of Factor Loadings by the Method of Maximum Likelihood," Procedures of the Royal Society of Edin., 1940, LX, 64-82.

Rao, C. R. "Estimation and Tests of Significance in Factor Analysis," Psychometrika, 1955, XX, 93,111.

II. Restrictions

Input is restricted to matrices of order 100 x 100 or less. Up to 50 factors can be handled by this program. If the number of factors decision is left to the program and more than 50 factors are estimated, an appropriate message will be printed out and control will be returned to the system.

III. Parameters

The program name is ITERATIVE FACTOR ANALYSIS. After the name on the call card the parameters must appear in the following order:

Parameter Number	Use or Meaning
1	Input Address of data matrix. CARDS or SEQUENTIAL 1-15.
2	Output Address of data matrix. SEQUENTIAL 1-15 and/or PRINT.
3	Output Address for principal axis factors. SEQUENTIAL 1-15 and/or PRINT.
4	Output Address of residual matrix. SEQUENTIAL 1-15 and/or PRINT.
5	Option code 0 if IPRAX 1 if ALPHA 2 if CANONICAL 3 if STEP-WISE MAXIMUM LIKELIHOOD

Parameter Number	Use or Meaning
6	Maximum number of cycles to be executed. If left blank, 50 cycles will be used as upper limit.
7	Number of factors to be extracted. If left blank, all factors with roots exceeding unity will be retained.
8	Exponent of convergence, n, where tolerance ETA = 10^{-n}. If left blank n = 3 or ETA = 10^{-3}. If all goes well, the program will stop as soon as either one of the stopping criteria is met. Error stops, if they occur, are labelled accordingly.
9	Sample size (for CFA only). If left blank, the chi-square computations are by-passed. If specified, chi-square is computed with the sampel size.
10	1 if input matrix was a covariance matrix.

A. Output common to all four sub-programs

1. Matrix output within system conventions:

   a. R (input covariance matrix)
   b. F (factor matrix)
   c. R-FF' (residual matrix)

2. Vector output, print only:

   a. communality vector (last iteration)
   b. vector of latent roots of C (last iteration)

3. Constant, print only:

   a. number of iterations completed
   b. largest discrepancy between trial vectors (H^2, U^{-1}, H^{-1}, depending on sub-program)
   c. root mean square of off-diagonal residual matrix
   d. per cent of variance removed

B. Additional output specific to sub-programs

AFA: The alpha-reliabilities of the m retained factors

CFA: chi-square statistic, chi-square appriximations (Wilson, Hilferty) for p = .05 and p = .01, for comparison with statistic. Degrees of freedom.

IV. Special Comments

The accuracy should be approximately 6 digits in computations, possibly somewhat lower for a very large number of iterations. The effective accuracy depends on the chosen tolerance ETA and the actual convergence as indicated by the largest discrepancy between trial vectors. The chi-square approximations are within 2×10^{-2} for more than 8 degrees of freedom.

JACOBI

I. General Description

This program calculates eigenvalues and eigenvectors of a square, symmetric matrix, using the JACOBI rotating technique. This program is limited to matrices of 110 rows and 110 columns. The user should realize that this technique is extremely slow on large matrices, while the program takes no longer than PRINCIPAL AXIS FACTOR ANALYSIS for small matrices (up to 20 x 20).

II. Parameters

Parameter Number	Use or Meaning
1	Input Address of correlation matrix. CARDS or SEQUENTIAL 1-15.
2	Output Address of eigenvectors.
3	Output Address of principal axis factor.
4	Output Address of eigenvalues, stored as a row vector. PRINT is not valid. The eigenvalues are always printed.
5	Number of eigenvectors (or factors) to be output.

III. Special Comments

The eigenvalues are stored in descending algebraic order (from largest to smallest), and the eigenvectors and factors are placed in the same order.

IV. Reference

Ralston, A. and Wilf, H. S.: Mathematical Methods for Digital Computers, John Wiley and Sons, New York, 1964.

OBLIMAX ROTATION

I. General Description

The OBLIMAX OBLIQUE ROTATION transforms a set of factors F to a new set V such that the factor kurtosis,

$$K = \frac{\Sigma\Sigma\, v_{ij}^{4}}{(\Sigma\Sigma\, v_{ij}^{2})^{2}} \qquad \begin{array}{l} i = 1, 2, \ldots\ldots\ldots, n \\ j = 1, 2, \ldots\ldots\ldots, k \end{array}$$

is at a maximum.

The purpose of the transformation is to attempt to rotate analytically to a position similar to that obtained by applying Thurstone's rules for simple structure. (See Multiple Factor Analysis, L. L. Thurstone, 1947, pp. 319-410.) However, Thurstone's rules and the oblimax procedure are not the same, and it is too much to expect that results obtained from both procedures will agree exactly.

It would be desirable to solve directly for the transformation matrix T, but unfortunately no solution to this problem has been found. Instead oblimax takes two vectors at a time, solves for the rotational angles, transforms the vectors, and then selects another pair until all k(k-1) pairs have been rotated. This process is repeated iteratively until the criterion K no longer increases. Despite the pairwise procedure, K is well behaved, and in general, approaches steadily to a minimum.

For any pair of factors, a and b, the solution proceeds as follows:

$$K = \frac{\Sigma\Sigma(a_i \cos \phi_j + b_i \sin \phi_j)^4}{[\Sigma\Sigma(a_i \cos \phi_j + b_i \sin \phi_j)^2]^2} = \frac{\Sigma\Sigma(a_i + b_i X_j)^4}{[\Sigma\Sigma(a_i + b_i X_j)^2]^2}$$

The derivative of K_{ab} is set equal to zero, resulting in a quartic equation in X which is tan ϕ. Two solutions for X will be maxima, and, each X is found, the sign of the second derivation is inspected to select maxima. A small transform (2 x 2) is created, but before post-multiplication is performed, the transforms must be adjusted so that when it becomes a part of T, t_a, and t_b, will remain normalized. In this way, both B and T are developed pair by pair.

For references see:

> Pinzka, C., and Saunders, D. R., "Analytic Rotation to Simple Structure II: Extension to an Oblique Solution." Research Bulletin RB-54-31. Princeton, N. J.: Educational Testing Service, 1954.

II. Alternate Use

If the user already has a transformation matrix, he may use it to compute V_{rs} et.al by giving the input address of T in parameter 8; in this case, the oblimax calculation of T will be skipped. If both F and T are to be input from cards, then the data deck of F should precede the deck of T.

III. Output

The OBLIMAX program always prints the following (unless parameter 8 is used):

1. The value of K for each pass

2. The iteration time

It outputs the following on demand (See Parameters):

1. Transformation matrix T

2. Reference vector structure, $V_{rs} = FT$

3. Reference vector correlations, $C_{rs} = T'T$

4. Diagonal of D and of D^{-1}
   where D is the diagonal matrix of the reciprocal square root of the diagonal elements of C_{rs}^{-1}

5. Primary factor pattern, $V_{fp} = FTD^{-1} = V_{rs}D^{-1}$

6. Primary factor correlations, $C_{fp} = DC_{rs}^{-1}D$

All data is printed out to seven decimal places.

IV. Restrictions

The number of variables plus the number of factors must be no more than 300.

V. Note on Parameter 2

If row normalization is specified, the normalization constants will be preserved and the rows will be rescaled to proper length after rotation and prior to output.

VI. Parameters

The program name, OBLIMAX, appears first on the program call card and is followed by the following parameters. Any output option (except parameter 9) may be SEQUENTIAL 1-15 and/or PRINT; it may be left blank if not desired.

Parameter Number	Use or Meaning
1	Input Address of F. CARDS or SEQUENTIAL 1-15.

Parameter Number	Use or Meaning
2	If rows are to be normalized before rotation, punch a 1; otherwise a zero or leave blank. (See Note on Parameter 2).
3	Output Address of T.
4	Output Address of V_{rs}
5	Output Address of C_{rs}
6	Output Address of V_{fp}
7	Output Address of C_{fp}
8	Input Address of T (See Alternate Use).
9	D-value and Inverse of D. PRINT only.

PRINCIPAL AXIS FACTOR ANALYSIS
(Eigenvalues and Vectors)

I. General Description

The purpose of PRINCIPAL AXIS FACTOR ANALYSIS is to determine a factor matrix, F, given a Gramian matrix, R, of order n such taht

$$F_{(n,f)}F'_{(f,n)} = R^*_{(n,n)}$$

where R* is an approximation to R.

The column vectors of F are defined as the factors (measures of dimensionality) of the original matrix, R. The solution for the matrix F is the classical eigen problem. Consequently, the computations are done by an eigenvalue subroutine. Before output the eigenvectors, E_j, are scaled as follows:

F(I,J) = E(I,J)*LAMBDA(J)**.5

for I = 1,....,n. J = 1,....,n.
to generate the principal axis factors, F. (See Introduction on Factor Analysis).

For a more detailed discussion see:

> Harry Harmon, Modern Factor Analysis, Chicago, University of Chicago Press, 1960, pp. 154-191.

II. Restrictions

The input matrix for the PRINCIPAL AXIS program must not exceed the dimensions of 190 x 190 double precision. The input matrix is further limited to being a square, symmetric matrix. Generally correlation, covariance, or cross-product matrices are used as input data. It should be noted that matrices with large numerical entries such as cross-products may generate output values which cannot be printed under the fixed output formats. The probability of this happening is very small. Any communality estimation (i.e., change in the diagonal entries of R) must be done prior to the input of R, to the PRINCIPAL AXIS program.

If the communality estimates are used, the user should check the resulting roots for negative numbers. If any exist the associated vector is meaningless.

The input data may come from any source conforming to SOUPAC. Similarly, the output codes follow the established conventions and are specified at the option of the user.

The R matrix may be completely factored (i.e., N factors from N variable matrix). However, there are three criteria which may be used to stop the factoring:

1. The user may specify the number of factors to be extracted. This criterion provides an upper limit beyond which factoring will not proceed. Therefore, it is necessary to put the maximum value in this limit in cases where it is not the primary criterion.

2. The percentage of total variance removed from R is the second limiting criterion. This parameter also specifies an upper limit to the process. Therefore, it should be set at 100 per cent unless it is the criterion for stopping.

3. The last criterion is to stop when the factor contribution (eigenvalue or root) falls below 1. The use of this procedure is dictated by the presence of its parameter.

If all three criteria are employed simultaneously, factoring is stopped by whichever criterion is first met.

III. Parameters

The parameters for the PRINCIPAL AXIS program appear on the program call card. They must follow the program name in this order:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15.
2	Output Adddres. SEQUENTIAL 1-15 and/or PRINT.
3	Maximum number of factors to be extracted. This must be less than or equal to the order of the input matrix.
4	The percentage of total variance to be removed expressed as an integer between 0 and 100.
5	The presence of a number greater than 0 indicates the factoring should stop when the eigenvalues (roots) fall below unity.
6	Output Address of Eigenvectors
7	The address of where eigenvalues are to be placed as a row vector if they must be stored for further use. If values need not be saved, leave parameter blank. PRINT is not valid.

Parameter Number	Use or Meaning
8	Mode of sorting eigenvalues and associated vectors. The codes are as follows:

Code	Meaning
0	Descending algebraic order
1	Descending absolute values
2	Order of extraction
10	Ascending algebraic order (the k smallest root)
11	Ascending absolute values
12	Reverse order of extraction

Leaving any parameter blank is the same as specifying zero. Consequently, options which are not needed can be avoided by leaving the associated parameter blank.

PROCRUSTES (Oblique Case)

I. General Description

This program offers 3 options:

1. (Oblique) Procrustes. Given A, B, the program solves

$$AT^* = B + E$$

for T^* in a least square sense (i.e., minimizing tr[E'E]), so that

$$T^* = (A'A)^{-1}A'B,$$

and then normalized T^* by columns to yield $T = T^*D$ so that diag (T'T) = I. It then computes AT which, in a loose sense, can be regarded as a least squares fit to A to B under the restriction that diag (T'T) = I. It also provided $C_f = D_n(T'T)^{-1}$ where D_n is a normalized diagonal matrix so that diag $(C_f) \doteq I$. If D gave the cosines between tests C_f will give the factor intercorrelations. A has to be a full column rank.

2. Dwyer Extension Analysis. Given $F = R_{tc}$, a centroid or equivalent matrix of cosines between tests t and uncorrelated factors c, and $L = R_{cn}$, a matrix of cosines between uncorrelated factors c and uncorrelated reference vectors n, this program computes

$$Q = T_{tn} = F(F'F)^{-1}L$$

which is used as a post-multiplier on some correlations matrix R_{et} between the tests t x in F and some set of extension variables e given R_{en}, the cosines of the extension variables e with reference n, to the extent that the former can be projected into the sub-space spanned by the latter. This multiplication

$$R_{en} = R_{et}\ T_{tn}$$

can be performed by use of the MATRIX program.

3. Left Inverse (transposed). Given A, the program will return

$$Q = A(A'A)^{-1}$$

provided A was a full column rank. Q is the transposed left inverse of A which can be used in lease squares application.

II. Restrictions

Input is restricted to matrices (A, B, or F) of order 190 x 50 or less.

III. Parameters

The parameters for this program appear on the program call card. They must follow the program name in this order:

Parameter Number	Use or Meaning	Procrustes	DEA	LINV
1	Input Address CARDS or SEQUENTIAL 1-15.	A	F	A
2	Input Address CARDS or SEQUENTIAL 1-15.	B	L	
3	Output Address SEQUENTIAL 1-15 and/or PRINT.	A	F	A
4	Output Address SEQUENTIAL 1-15 and/or PRINT.	B	L	
5	Output Address SEQUENTIAL 1-15 and/or PRINT.	T	Q	$A(A'A)^{-1}$
6	Output Address SEQUENTIAL 1-15 and/or PRINT.	C_f		
7	Output Address SEQUENTIAL 1-15 and/or PRINT.	AT		
8	Output Address SEQUENTIAL 1-15 and/or PRINT.	E		
9	Choice Address	0	1	2

SQUARE ROOT FACTOR ANALYSIS

I. General Description

The SQUARE ROOT method of factor analysis, also called the Diagonal Method, by L. L. Thurstone, decomposes a correlation matrix R (or any other positive semi-definite or definite symmetric matrix) such that

$$R = F\,F' + R(k+1)$$

where R(k+1) is the residual matrix after extracting k factors. Of course if all n factors are extracted, the residual matrix becomes a null matrix.

The factor f_j is computed by dividing each element of the j^{th} column of R by its diagonal square root:

$$f_{ij} = r_{ij}/\sqrt{r_{jj}} \quad (i = 1,2,\ldots\ldots,n)$$

The matrix $A = f_j \cdot f'$ is then subtracted from R and the operation repeated on the residual matrix.

Prior to the widespread use of high speed computers, the SQUARE ROOT method was sometimes used as a substitute for the PRINCIPAL AXIS or CENTROID method due to the relative ease of computing a square root factor. When used in this way, one seeks to extract the maximum variance for each factor, in which case Parameter 4 should be blank. The program then selects the next column on the basis of the largest residual column sum of squares.

Nowadays, however, the SQUARE ROOT method is more likely to be used for special purposes. By selecting successive pivot variables, the user retains control over the factoring. Factors are passed directly through the test variables and the effect of these variables is removed from the matrix. The communalities or row sums of squares are the squared multiple correlations of the remaining variables with the pivot variables.

The pivots selected may be any columns in the matrix. Let us assume, however, that these are adjacent to each other in the upper right hand corner of the partitioned matrix below:

$$R = \begin{bmatrix} R_{pp} & R_{ps} \\ R_{sp} & R_{ss} \end{bmatrix}$$

Then the effect of pivoting successively on the variables in the upper right hand corner is shown by the residual matrix as follows:

$$R^{(p+1)} = R - F_r F_p' = \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & R_{ss} - R_{sp} R_{pp}^{-1} R_{ps} \end{array}\right)$$

II. Restrictions

A. Dimension

Maximum size of the R matrix is 190 variables.

B. Special Conditions

1. The researcher may specify the extraction of any number of factors up to dimension of R.

2. The researcher may specify the diagonal element to be used in the extraction of each factor, or he may have the procedure remove the maximum variance each time.

3. The residual matrix may be saved if the researcher desires.

III. Parameters

Following the program name the parameters must appear in the following order on the program call card:

Parameter Number	Use or Meaning
1	Input Address. CARDS or SEQUENTIAL 1-15. (Correlation or positive definite or semi-definite matrix).
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.
3	Number of factors extracted.
4	Input Address for diagonal elements. CARDS or SEQUENTIAL 1-15.
5	Output Address for residual matrix. SEQUENTIAL 1-15 and/or PRINT.

IV. Special Comments

If the diagonal element for each factor is specified, and if both input addresses are cards, then data precedes diagonal specification.

V. Example

Assume you have a 20 x 20 correlation matrix on cards and that you want to extract 15 factors; also you are reading the pivot column from cards. The program would be set up as follows:

```
/*ID
//  EXEC  SOUPAC
//SYSIN  DD  *
SQU(C)(P)(15)(C)(P).
END SOUPAC
DATA(20)(8F9.7)
      .
      .    data
      .
      .
END #
DATA(15)(15I2)
.................................. diagonal specification card(s)
END #
/*
```

THREE-MODE FACTOR ANALYSIS

I. General Description

A. GENERAL COMMENTS

This program provides a factor analytic solution for a 3-dimensional i by j by k data matrix. The computational procedures employed are those presented in Method III of Tucker's article (reference below). This method provides most efficient analysis when one of the modes, usually individuals is quite large, though this is certainly not a necessary condition.

B. THE THEORETICAL MODEL

Here, i, j, and k represent the modes of classification which are directly related to the observation of the data; i, j, and k are thus termed observational modes. An example would be the observation of scores for i individuals on j tests given under k different conditions.

Through factoring, we wish to reduce the observational modes i,j, and k to corresponding derivational modes m, p, and q. Each of the derivational modes can be thought of as a set of factors in the domain of the corresponding observational mode. The core matrix G then serves to describe the relationships among the derivational modes.

The fundamental three-mode factor analysis model is represented by the equation:

$$\tilde{x}_{ijk} = \sum_m \sum_q \sum_p a_{im} \, b_{jp} \, c_{kq} \, g_{mpq} \quad ,$$

where $\tilde{x}_{ijk}$ is an approximation to the observed score x_{ijk} ; a_{im}, b_{jp}, and c_{kq} are entries in two-mode matrices ${}_iA_m$, ${}_jB_p$, and ${}_kC_q$ describing the elements in the observational modes i, j, and k in terms of the dimensions in the derivational modes m, p, and q respectively; the coefficients g_{mpq} are entries in a three-dimensional matrix G and represent the measures of the phenomenon being observed for each combination of the dimensions of the derivational modes.

In matrix form, the model could be represented as:

$${}_i\tilde{X}_{jk} = {}_iA_m \, G_{(pq)} \left({}_pB_j \otimes {}_qC_k \right) \quad ,$$

where $\otimes$ indicates a Kronecker product. Matrices A, B, and C are factor solutions for modes i, j, and k respectively which serve to transform the core matrix G of the 3 derivational modes to the matrix X representing the 3 observational modes.

C. INPUT DATA

The input data must be a Gramian matrix, usually correlations, covariances, or cross-products, in the form ${}_{jk}R_{jk}$, where i is assumed to be the largest mode and jk represents the combination mode, with mode k nested within mode j.

II. Output

The output consists of the following:

(1) the ${}_{j}P_{j}$ and ${}_{k}Q_{k}$ matrices which represent the correlations, covariances, or cross products within modes j and k respectively;

(2) the eigenvalues and eigenvectors of ${}_{jk}R_{jk}$;

(3) the eigenvalues and eigenvectors of ${}_{j}P_{j}$;

(4) the eigenvalues and eigenvectors of ${}_{k}Q_{k}$;

(5) the core matrix ${}_{pq}G_{m}$, where m, p, and q represent the derivational modes corresponding to observational modes i, j, and k respectively.

All of this is printed out and may also be stored on sequential storage devices. The user must specify the number of factors to be extracted from each of the three modes. This procedure is employed since the use of other factor-stopping criteria (e.g. per-cent of variance accounted for, or eigenvalues below unity) could easily lead to the computation of a great many useless factors as well as a very large and unmanageable core matrix. The user is also cautioned against specifying large numbers of factors since this would cause substantial increases in time required to factor the various modes and compute the core matrix.

It is strongly recommended that the user be familiar with the Tucker article and with factor analysis in general before attempting to use this program.

III. Parameters

The program mnemonic (T-M) appears first on the program card and is followed by the following 16 parameters, the first 8 of which are required. Output addresses are optional. All output will be printed.

Parameter Number	Use or Meaning
1	Input address of R matrix in the form ${}_{jk}R_{jk}$ (Cards or S1-S15).
2	Number of subjects or elements in mode i.
3	Number of variables in mode j.

Parameter Number	Use or Meaning
4	Number of variables in mode k.
5	Number of factors to be removed from matrix R (mode i).
6	Number of factors to be removed from matrix P (mode j).
7	Number of factors to be removed from matrix Q (mode k).
8	(1) If input matrix is cross-products (2) If input matrix is correlations. (3) If input matrix is covariances.
9	Output address for row vector of eigenvalues of R.
10	Output address for eigenvectors of R.
11	Output address for matrix ${}_jP_j$.
12	Output address for row vector of eigenvalues of P.
13	Output address for eigenvectors of P.
14	Output address for matrix ${}_kQ_k$.
15	Output address for row vector of eigenvalues of Q.
16	Output address for eigenvectors of Q.
17	Output address for core matrix ${}_{pq}G_m$.

IV. Reference

Tucker, Ledyard R. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 1966, 31, 279-311.

UNRESTRICTED MAXIMUM LIKELIHOOD FACTOR ANALYSIS

Parameter Number	Use or Meaning
1	Input Address for correlation matrix. SEQUENTIAL 1-15; CARDS are not permitted
2	Output Address for final unrotated factor matrix. SEQUENTIAL 1-15. See also Parameter 13.
3	Input Address for row vector of initial estimate of uniqueness. CARDS, SEQUENTIAL 1-15 (optional).
4	Lower bound for number of factors.
5	Upper bound for number of factors.
6	Sample size (number of observations) on which correlation matrix is based.
7	Maximum number of iterations.
8	Probability of chance occurance, i.e., *1.00*.
9	1 to print input correlation matrix and partial correlation matrices after any variables have been removed.
10	1 to print technical output.
11	1 to print intermediate results.
12	1 to punch unrotated factor matrices.
13	1 to apply a varimax rotation to all factor matrices. If this parameter is used the output of parameter 2 will be a rotated factor matrix.

This program has been taken directly from Jöreskog (1967) with his permission. Anyone interested in the methods is referred to the references listed below. The program is temporarily limited to 75 variables and 30 factors. Parameters 1, 4, 5, 6, 7, and 8 are required. Parameter 8 must be enclosed within asterisks, **, and must have a punched decimal point.

References:

Jöreskog, K. G. UMLFA - a computer program for unrestricted maximum likelihood factor analysis. Research Memorandum 66-20. Princeton, New Jersey: Educational Testing Service. Revised Edition, 1967.

Jöreskog, K. G. Some contributions to maximum likelihood factor analysis. Psychometrika, 1967, 32, 443-482.

VARIMAX FACTOR ROTATION

I. General Description

VARIMAX ROTATION is used to redistribute a factor matrix (principal axis, centroid, etc.) variance so that the matrix approches orthogonal simple structure. The varimax scheme maximizes the following criterion function:

$$\sum_{s} \left(h\sum_{j}(a_{(j,s)}{}^{2}/h_{(j)}{}^{2})^{2} - (\sum_{j}(a_{(j,s)}{}^{2}/h_{(j)}{}^{2}))^{2}\right)$$

where j is the variable index number: 1,........, n

s is the factor index number: 1,........, f

$a_{(j,s)}$ is the factor loading of the j^{th} variable on the s^{th} factor

$h_j{}^2$ is the j^{th} variable communality

For further discussion see:

H.F. Kaiser, "Computer Program for Varimax Rotation in Factor Analysis", Educational and Psychological Measurement, Vol. XIX, Nov. 3, 1959, pp.413-420.

Cooley and Lohnes, Multivariate Procedures for the Behavioral Sciences, New York, John Wiley and Sons, Inc., 1962, pp.161-3.

II. Restrictions

The input matrix for VARIMAX ROTATION must not exceed 190 variables and 190 factors. The number of factors may be anything greater than or equal to 2. Any factor matrix generated by a statistical system factor analysis program is acceptable input. A matrix may also be entered from cards.

III. Parameters

The parameters for the VARIMAX ROTATION appear on the program call card. They must follow the program name in this order:

Parameter Number	Use or Meaning
1	Input Address. CARDS* or SEQUENTIAL 1-15.
2	Output Address. SEQUENTIAL 1-15 and/or PRINT.
3	The presence of a number greater than 0 in this parameter indicated the communalities should be printed.
4	0 or blank for normal VARIMAX. 1 if raw VARIMAX is desired.

*If CARDS are used the DATA card must contain the number of rows as well as the number of columns in the input matrix (see User's Guide for details).

SOUPAC (Statistically Oriented Users Programming and Consulting)

ECONOMETRICS PACKAGE

ECONOMETRIC REDUCED FORM AND RESIDUAL ANALYSIS

I. General Description

The ECONOMETRIC REDUCED FORM AND RESIDUAL ANALYSIS program calculates the following:

(1) Residuals

$$U = (YX) \begin{Bmatrix} \beta \\ \Gamma \end{Bmatrix}'$$

where:

(YX) is the raw data matrix of endogenous and exogenous variables and $\begin{Bmatrix} \beta \\ \Gamma \end{Bmatrix}$ is the matrix of coefficient estimates.

(2) Durbin-Watson statistic each equation(i)

$$\frac{\sum_{t=2}^{N} [U_i(t) - U_i(t-1)]^2}{\sum_{t=1} [U_i(t)]^2}$$

(3) Covariance matrix for residuals

$$W = \begin{Bmatrix} \beta \\ \Gamma \end{Bmatrix} (S) \begin{Bmatrix} \beta \\ \Gamma \end{Bmatrix}'$$

where:

S is the raw data covariance matrix.

(4) Reduced form estimates

$$\pi = -(\beta)^{-1}(\Gamma)$$

(5) Reduced form predicted values

$$\tilde{Y} = Y\pi$$

(6) Reduced form residuals

$$V = Y - \tilde{Y}$$

(7) Covariance matrix for reduced form residuals

$$(\beta^{-1})' W(\beta^{-1})$$

References:

Johnston, J., Econometric Methods, New York, McGraw-Hill Book Company, Inc., 1960.

Goldberger, Arthur S., Econometric Theory, New York, John Wiley and Sons, Inc., 1964.

II. Restrictions

Only those inputs used in the calculations called for need be given. They must be in the following formats:

(1) Coefficients:

The coefficient matrix for K equations with N variables, N1 exogenous and N2 endogenous, must be a K by N+1 matrix. Each row corresponds to an equation. The first element in each row is the constant term followed by the coefficients matrix (i.e., exogenous coefficients first; endogenous coefficients next). In each row, there must be -1 which corresponds to the endogenous variable that was normalized on.

(2) Raw Data:

The data must be arranged so that exogenous variables occur first and endogenous variables last. (The TRANSFORMATION program may be used to arrange data in this way, if it is not already like this).

(3) Raw Data Covariance Matrix:

The covariance matrix must have the following form

sample size	$\bar{X}$ (means)
σ (standard deviations)	Covariance (exogenous first, endogenous last)

Care should be taken to see that an input address is specified for any data needed in calculating the desired statistics and that any intermediate statistics needed are stored (i.e., an output address besides print is specified). The following list indicates which previous statistics are needed in the calculation of each statistic.

1. Residuals - coefficients and raw data

2. Dubrin-Watson statistic - coefficients and raw data

3. Covariance matrix for residuals - coefficients and raw data covariance matrix

4. Reduced form coefficients - coefficients

5. Reduced form predicted values and residuals - reduced form coefficients (no output address) and raw data

6. Covariance matrix of reduced form residuals - reduced form coefficients and covariance matrix of orginal residuals

This program is restricted to less than 150 variables.

III. Parameters

The parameters appear on the program card following the name ECONOMETRIC in the following order:

Parameter Number	Use or Meaning
1	Input Address for coefficients. SEQUENTIAL 1-15.
2	Input Address for raw data covariance matrix. SEQUENTIAL 1-15.
3	Input Address for raw data. SEQUENTIAL 1-15. (See Special Comments).
4	Number of exogenous variables (total).
5	Output Address for residuals. SEQUENTIAL 1-15 and/or PRINT.
6	Output Address for covariance matrix of residuals. SEQUENTIAL 1-15 and/or PRINT.
7	If greater than 0, reduced forms are calculated and printed.
8	If greater than 0, reduced form predicted values and residuals are printed.
9	If greater than 0, covariance matrix for reduced form residuals is printed.

IV. Special Comments

If Parameter Number 3 is specified, the Durbin-Watson statistic will be calculated and printed.

K-CLASS ESTIMATION

I. General Description

K1Class and K2Class programs work together to do K-class estimation. There are three estimators which belong to the K-class. These include: Ordinary Least Squares (multiple regression), Two-stage Least Squares, and Limited Information Maximum Likelihood.

II. Description of K1Class Output

The K1Class program calculates the basic statistics:

Mean: $\bar{X}_i = \frac{\Sigma X_i}{N}$ where N = Sample Size

Variance Covariance: $S_{ij} = \frac{\Sigma X_i X_j}{N} - \bar{X}_i \bar{X}_j$

Standard Deviation: $s_i = \sqrt{S_{ii}}$

Correlation: $C_{ij} = \frac{S_{ij}}{s_i s_j}$

Cross-products in matrix notation: $CP = X'X$

K1Class also calculates the eigenvalue to be used in Limited Information Maximum Likelihood estimation in K2Class.

Variables must be ordered; exogenous, endogenous, (independent, dependent).

III. Description of K2Class Output

K2Class calculates estimates and associates statistics for Ordinary Least Squares (OLS), Limited Information Maximum Likelihood, and Two-stage Least Squares.

Formulas

$\hat{\beta}'$ $\equiv$ estimates of the endogenous variables

$\hat{\gamma}$ $\equiv$ estimates of the exogenous variables

y_o $\equiv$ variable normalized on

Y $\equiv$ endogenous variable in the equation

X_* $\equiv$ exogenous variables in the equation

X $\equiv$ exogenous variables in the system

Estimating Formulas

$$\begin{bmatrix} \hat{\beta} \\ \hat{\gamma} \end{bmatrix} = - \begin{bmatrix} Y'Y - kV'V & Y'X_* \\ X_*'Y & X_*'X_* \end{bmatrix}^{-1} \begin{bmatrix} Y'y_o - kV'y_o \\ X_*'y_o \end{bmatrix}$$

or

$$\Theta = A^{-1} J$$

where $V'V = Y'Y - Y'X(X'X)^{-1}X'Y$

$V'y_o = Y'y_o - Y'X(X'X)^{-1}X'y_o$

k determines the estimating technique.

Standard Error of Estimate

$$\hat{\sigma} = \sqrt{\frac{y_o'y_o - (\hat{\beta}\ \hat{\gamma})\ C}{N-j}}$$

where j = Rank of A^{-1}

Standard Error of the Estimated Coefficients.

s^*_{ii} = The square root of the i th diagonal element of the $\hat{\sigma}^2 A^{-1}$ matrix

t-ratio:

$$\frac{\hat{\theta}_i}{s^*_{ii}}$$

Covariance matrix of the coefficients:

$$C^2 = \sigma^2 A^{-1}$$

IV. Ordinary Least Squares (OLS)

When K = 0, the K-class formula reduces to Ordinary Least Squares. Y is then assumed to be another exogenous variable in the equation. When ordinary least squares is specified the following additional statistics are supplied.

In matrix notation

$$R^2 = \frac{\hat{\Theta}C - \Sigma y_o^2 / N}{y_o'y_o - \Sigma y_o^2 / N} = \frac{\text{Explained Sum of Squares}}{\text{Total Sum of Squares}}$$

Total Sum of Squares

$$TSS = y_o' \, y_o - \Sigma^2 y_o / N$$

Regression (Explained) Sum of Squares:

$$RSS = \hat{\Theta}C - \Sigma y_o^2 / N$$

Error (Unexplained) Sum of Squares:

$$ESS = TSS - RSS = y_o'y_o - \hat{\Theta}C$$

Example: Suppose that there were two possible models that one wanted to estimate. In one model variable 10 is included and in the other variable 10 was not included.

```
K1C(S1)(S2)(0)( )( )(1)(1)( )(1).
END P
K2C(S2)(S3)(S4)( )(10)(2)*0*.
(10)(1)(1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11).
(9)(1)(1)(2)(3)(4)(5)(6)(7)(8)(9)(11).
END P
```

V. Limited Information Maximum Likelihood (LIML)

When K = the smallest eigenvalue as calculated in K1Class, K2Class calculates Limited Information Maximum Likelihood estimates.

The eigenvalue is calculated in the following manner:

Let

$$W_* = Y'Y - Y'X_* \, (X'X)^{-1} X_*' \, Y$$

$$W = Y'Y - Y'X(X'X)^{-1} \, X'Y$$

K1Class transfers to K2Class the smallest eigenvalue of the matrix $W^{-1}W_*$.

Example: Suppose there is a two equation system with four exogenous and two endogenous variables. The following program will calculate LIML estimates for both equations:

```
K1C(C)(S1)(0)(2)(4)( )( )(S2).
(2)(2)(1)(4)(5)(6).
(2)(2)(2)(3)(5)(6).
END P
K2C(S1)(S3)(S4)(S2)(4)(2)*-1*.
(2)(2)(1)(4)(5)(6).
(2)(2)(2)(3)(5)(6).
END P
```

VI. Two-stage Least Squares(2SLS)

When K = 1 is specified Two stage Least Squares estimates are computed.

Example: Suppose one has a three equation model that contains three endogenous and ten exogenous variables. The following program will calculate 2SLS estimates.

```
K1C(C)(S2).
END P
K2C(S2)(S3)(S4)( )(10)(3)*1.*.
(5)(2)(1)(2)(3)(4)(5)(11)(13).
(4)(2)(3)(4)(6)(7)(12)(13).
(5)(2)(1))2)(8)(9)(10)(11)(12).
END P.
```

VII. Parameters

A. MAIN Parameters

1. K1Class

Parameter Number	Use or Meaning
1	Input Address. CARDS, SEQUENTIAL 1-15.
2	Output Address for raw data covariance matrix. SEQUENTIAL 1-15.
3	Type of Input 0 = raw data 1 = cross-products 2 = covariance
4	Number of equations of LIE; 0 otherwise
5	Total number of exogenous variables in LIE system; $1 \leq n \leq 94$.
6	1 if want covariance matrix printed.

Parameter Number	Use or Meaning
7	1 if want cross-products matrix printed.
8	Output Address for eigenvalues. SEQUENTIAL 1-15.
9	1 if want correlations matrix printed.

2. K2Class

Parameter Number	Use or Meaning
1	Input Address same as output address for K1Class.
2	Output Address for estimated coefficients. SEQUENTIAL 1-15.
3	Scratch Address SEQUENTIAL 1-15.
4	Input Address for Eigenvalues same as output address in K1Class.
5	Number of exogenous variables in the system.
6	Number of equations.
7	Floating point value of k. (See special comments.) This value should be enclosed in asterisks.

B. Sub-parameters (Equation Control Cards)

Subparameters are needed for all K2Class programs and for K1Class when LIML estimates are desired. There is a subparameter card for each equation. If subparameters appear in both K1Class and K2Class, they must appear in the same order.

Each equation control card has the following form:

Parameter Number	Use or Meaning
1	Number of exogenous variables in the equation.
2	Number of endogenous variables in the equation.
3	The variable number of all variables in the equation in the order:

1 - exogenous in the equation
2 - endogenous in the equation with
variable standardized on last.

VIII. Special Comments

A. If K = *0* Ordinary Least Squares Estimates are computed.
If K = *1* 2-Stage Least Squares Estimates are computed.
If K = *-1* Limited Information Maximum Likelihood Estimation are computed.

B. When LIML estimates are desired, the output and input addresses for eigenvalues must be specified .

C. K1Class accepts data from cards or intermediate storage either as raw data, cross-products, or covariance. If cross-products or covariance are used as input to K1Class, the matrix must be in the following order:

Cross-products

N	Σx
Σx	cross-products

Covariance

N	$\bar{X}$
σ	covariance

Input to K2Class must come from K1Class. However, once data has been processed by K1Class, it may be used by K2Class any number of times.

IX. References

Goldberger, A. S., Econometric Theory, New York, John Wiley and Sons, Inc., 1964.

Johnston, J., Econometric Methods, New York, McGraw-Hill Book Company, Inc., 1960.

LINEAR PROGRAMMING

I. General Description

LINEAR PROGRAMMING maximizes or minimizes a linear function subject to certain linear inequalities called constraints.

In matrix notation:
Find the solution to
$AX <, =, > b$ (a system of linear equations or inequalities)
which maximizes (or minimizes)
$Z = CX$
where $X \geq 0$

A is the matrix of coefficients of the constraints, X the vector of variables, C the vector of costs or profits associated with each variable, and b a vector or matrix of non-negative constants which places a bound on the linear equations.

The equations, $AX <, =, > b$ in n variables define and bound a space called the feasible space in which all allowable values of the n variables are defined. The SIMPLEX criterion finds those combinations of variables which optimize the objective function within this feasible space. To solve the system of linear equations defined above, the inequalities must be changed to equalities. This is accomplished by addition of surplus variables to "greater than" constraints, and slack variables to "less than" constraints. To create the basis for solving a system of linear equations, an identity matrix must be formed and augmented to the A matrix of structural variables. Creation of the identity matrix is completed by addition of artificial variables to constraints with a "greater than" relational operator. The program adds any needed variables.

Since there are more variables (structural + surplus + slack + artificial) than rows, some method must select which variables will be in solution. The SIMPLEX Algorithm selects a number of variables (equal to the number of rows) which will be in solution. The final solution is the maximum (or minimum) of the linear function subject to the constraints. Since slack and surplus variables have "real" meaning, they may appear in the final and intermediate solutions. Their presence as a non-zero value indicates that the constraint to which they were added is not binding. Artificial variables have no "real" meaning. Presence of artificial variables in solution indicates that some constraints are so constructed as to preclude a solution which has "real" meaning.

The slack and surplus variables are given costs of zero in the objective function. Artificial variables are given large negative costs. SIMPLEX attempts to drive artificial variables from solution.

Failure to drive artificial variables from solution may indicate a problem in which constraints are mutually exclusive or that the cost assigned to the artificial variable is not large enough.

In matrix notation the augmented matrix before calculations begin would appear as:

$$A \mid I \mid S \mid = b$$

where I is the identity matrix of slack and artificial variables and S is the matrix of surplus variables. Row operations are performed on the augmented matrix according to the SIMPLEX criterion. After any number of row operations, the inverse matrix of the original coefficients of structural variables now in solution is contained in the columns where the original identity matrix was located. At every stage (row operation) an identity matrix will be present. This identity matrix indicates the variables in solution.

Since the original table is stored by the program, it is possible to compare the results of the inverse obtained through LINEAR PROGRAMMING with the inverse obtained by a standard inversion technique. The user may set the absolute value for this comparison in Parameter 3. If the comparison does not meet the accuracy requirement, a new table is formed using the original table and the calculated inverse. After a feasibility check, the program continues calculations until an acceptable solution is obtained.

References

Llewellyn, R. LINEAR PROGRAMMING. New York, New York: Holt, Rinehart, and Winston, 1966.

Hadley, G. LINEAR PROGRAMMING. Reading Massachusetts: Addison-Wesley, 1963.

II. Restrictions

The program is limited to a maximum of 90 rows or constraints, 300 columns or variables, and 5 columns in the requirement matrix.

These limits are internal limits and the user is warned that large problems may exceed the program capacity during accuracy check and calculations involving multiple column requirement matrices. Program capacity WILL be exceeded if: the number of constraints + number of structural variables + number of "greater than" inequalities > 300.

Input may come ONLY from CARDS in the form of subparameters.

III. Parameters

All floating point numbers (indicated by FP) must be enclosed by a pair of asterisks. All integer numbers (indicated by IN) must be enclosed in parentheses. The main call to the program and each subparameter must be terminated by a period (.).

The program is entered by punching the symbols L-P followed by the appropriate main parameters and subparameters. All main parameters have default options.

Main Parameters to follow L-P

1 Cost of artificial variables *large negative FP numbers*. Default = -1.E50.

2 Minimum value for calculations *FP*. If any calculation falls below this value, it is set to zero. Default = internal calculations.

3 Value for accuracy check *FP*. If absolute value for calculated difference (See General Description) falls below this value, final value is termed inaccurate and calculations are performed to correct rounding errors. Default = .5 .

4 If 1, suppress print of solution matrix (IN).

5 If 1, suppress print of check matrix (IN).

6 Print every INth step, i.e. row operation (IN).

7 If 1, insert small positive, non-zero number for any zero in the b vector. Useful aid if b vector contains many zeros.

Subparameter

The program now expects to find the word MINimize or MAXimize followed by a string of constants which represent, in sequential order, the cost or values associated with each variable. All non-zero constants (with or without decimal) must be enclosed by a pair of asterisks. Zeros may be enclosed by asterisks. A series of sequential zeros may be represented by a pair of parentheses, i.e. the integer number in the pair of parentheses represents the number of sequential zeros to be inserted. All coefficients must appear and be in sequential order.

The cost coefficients representing the objective function are terminated by a period. The constraints are entered in a similar manner. All variables must be in sequence. Coefficients of zero must be included. Multiple requirement vectors are entered in the standard form. The constraint is terminated by a period. Comments which do not include period (.), comma (,), asterisks (*) or left parenthesis may be entered at any point outside those characters delimiting constants. The requirement vectors are separated from the rest of the constraint by relational operators. All coefficients must appear and be in sequential order.

The program recognizes three relational operators: LE (less than or equal), EQ (equal), and GE (greater than or equal). These relational operators are surrounded by quotes ("). See Section V. Examples in this program.

Output

The output consists of the objective function, the final solution matrix, the variables in solution, and the optimal functional value. In addition, Shadow Prices or opportunity costs are printed. Shadow Prices provide useful information on the "cost" of having certain constraints, or the increased profit to be obtained by 'relaxing' a particular constraint.

For example:

Constraint 1: $1X(1) + 2X(3) \leq 5.0$

To this constraint, slack variable X(I) is added to make it an equality. In the final solution, X(I) is not in solution. The optimal maximum functional value is 20. The 'Shadow Price' on variable X(I) is 2.0. This means that if we relax this constraint to 6.0, the optimal maximum value could be 22.0. For every unit the constraint is relaxed, the functional value will be changed by the Shadow Price. The Shadow Price holds until the constraint is no longer binding. The same logic may be applied to "GE" type constraints with surplus variables. For interpretation of Shadow Price for structural and artificial variables, the user is referred to texts under headings such as "Dual Algorithm", Interpretation of the Dual", and "Opportunity Costs".

Basis variables refer to those variables which form the original identity matrix. The variable numbers are listed in the order they were added to the constraints. The number of basis variables will always equal the number of constraints. To determine whether a basis variable is a slack or artificial variable, refer to the coefficients of these variables in the objective function. A slack variable will have a coefficient of 0.0.

MESSAGES

PROBLEM TOO LARGE: More than 300 variables or 100 constraints on input or during addition of slack, surplus, and artificial variables.

NORM FOR CUTOFF: Value of Main Parameter Number 2, either supplied or default.

ERROR IN SIMPLX: Source Program Error. See a consultant.

SOLUTION UNBOUNDED: Constraints do not form a closed space. Optimal functional value is infinite.

NUMBER OF ITERATIONS: Number of row operations needed to calculate final solution. For multiple requirement vectors, number is not cumulative.

ACCURACY ACCEPTABLE or ACCURACY NOT ACCEPTABLE: Comparison with Main Parameter Number 3.

VARIABLE ADDED

NEW BASIS VARIABLES ARE: Iterations either inaccurate and new variable added or, during execution of multiple requirement vector, a new variable had to be added to make problem feasible (requirement vector positive).

NON-RESOLVABLE TIE: Cannot occur mathematically. Only reason for occurance is due to rounding error in machine. Can be corrected by incrementing or decrementing requirement vector by a small amount. Perform this only for constants of same value. (Use Parameter 7).

Other messages should be self-explanatory.

Special Comments

Speed and accuracy can be increased by observing the following suggestions:

1) Never make Parameter 3 (accuracy check) larger than (0.1) X (number of significant digits in table). For example, if numbers in the table are 4, 5, .001, 8c, 95.32, you have "one" significant digit. Set Parameter 3 to ".1".

2) Scale numbers in table to get them into same range. For example, if table entries are of the order 10^1, and the requirement vectors are of the order 10^3, scale requirement vectors to 10^1 and rescale solution by 10^2. The objective function may also be rescaled in a similar manner. Rescaling essentially reflects the number of significant digits.

V. Examples

The problem:

Minimize $-.75X(1) + 150X(2) -.02X(3) + 6X(4)$

Subject to the following constraints:

Constraint(1)

$.25X(1) -60X(2) -.04X(3) + 9X(4) \leq 0, 1, 2$

Constraint(2)

$.05X(1) -90X(2) -.02X(3) -3X(4) \leq 0, 1, 2$

Constraint(3)

$1X(3) \leq 1, 2, 3$

Could be set up on cards as follows:

```
/*ID
//  EXEC  SOUPAC
//SOUPAC.SYSIN  DD  *
L-P*-1.E20****.1*()()(1).
MIN*-.75**150**-.02**6*.
LABOR *.25**-60**-.04**9*"LE"*0**1**2*.
LAND*.05**-90**-.02**-3.0*"LE"*0**1**2*.
CASH(2)*1*(1)"LE"*1**2**3*.
END PROGRAM
END SOUPAC
/*
```

This problem will result in an unbounded solution with requirement vector number one. The problem terminates without performing calculations on the other vectors.

Note insertion of sequential zeros on CASH card.

QUADRATIC PROGRAMMING

I. General Description

This program maximizes the quadratic function $cx + 1/2\ x^T Dx$ subject to the linear constraints $Ax \leq b$, where c is an n-vector, D is a symmetric negative definite n by n matrix, A is an m by n matrix of coefficients or constraints and b is an m vector.

The Kuhn-Tucker theory shows that a solution to the constrained maximization problem is obtained if and only if vectors x, L, v, and w can be found such that:

$$\begin{array}{l} \underline{Dx - A^T L + v = -c} \\ Ax \qquad\quad + w = \ \ b \end{array}$$

where the elements of x, L, v, and w are non-negative and the conditions $xv = 0$ and $lw = 0$ are satisfied. To find these vectors, artificial vectors z^1 and z^2 are added to the first equation and a y-vector is added to the second. Simples techniques are then used to eliminate first the y and then the z^1 amd z^2 variables.

References:

Carr, C. R. and C. H. Howe, Quantitative Decision Procedures in Management and Economics, McGraw-Hill, 1964.

Hadley, G., Nonlinear and Dynamic Programming, Addison-Wesley, 1964.

Wolfe, P., "The Simplex Method for Quadratic Programming", Econometrica, 27, 1959, pp. 382-398.

NOTE: Carr and Howe claim that elements of the w-vector may not be en entered in the first stage of the simplex procedure. Since this requires that there exist a solution to $Ax = b$, it is a severe restriction. It is also unnecessary, and this program does enter w-variables during the first stage. Otherwise, the procedures used closely follow those of Carr and Howe.

II. Restrictions

The maximum number of x-variables is 40. The number of x-variables plus the number of constraints must be $\leq$ 80.

The D-matrix must be negative definite. If this is dubious, use the PRINCIPLE AXIS FACTOR ANALYSIS program to extract the eigenvalues. All must be negative. Semi-definite D-matrices may be perturbed or the user may limit the number of iterations to be performed. If this limit is exhausted, final solution vectors will be printed out (see below).

The only form of input is a matrix of data. If there are n x-variables and m constraints, the matrix should have n + 1 columns and m + n rows, partitioned as follows:

$$\begin{array}{c|c} D\ (n \times n) & c\ (n \times 1) \\ \hline A\ (m \times n) & b\ (m \times 1) \end{array}$$

Note that this is the c vector, not the -c vector mentioned in the Kuhn-Tucker formulas. Also note the + sign and the 1/2 coefficient of the xDx term. All constraints in this type of input are assumed to be $\leq$ type. Multiply $\geq$ constraints through by -1. The equality constraing:

$$\sum_{j=1}^{n} a_{ij}X_j = b_i$$

is equivalent to the two constraints $\Sigma a_{ij}X_j \leq b_i$ and $\Sigma -a_{ij}X_j \leq b_i$.

This matrix can be read in from cards or from temporary storage. Only one problem can be read from a tape or sequential location. Multiple problems must be read from cards, the matrix for each problem preceded by its own "DATA (N + 1) (FORMAT)" card and followed by its own "END#" card.

The elements of the w-vector are always non-negative and are to be considered "slack" for $\leq$ constraints and "surplus" for $\geq$ constraints.

The user may obtain the basis vector at the end of each iteration showing which variables are in the basis and their quantities (option 2). He may alternatively have the entire matrix printed out after each iteration (option 3). The user is cautioned that option 3 can use immense quantities of paper and time unless the problem is very small.

A method outlined in Hadley, pages 183 - 186, is used to avoid cycling in cases of degeneracy.

. Parameters

The program call card should have the name QUADRATIC PROGRAMMING followed by these parameters:

Parameter Number	Use or Meaning
1	Number of problems following
2	Input Address. SEQUENTIAL 1-15 or CARDS.
3	Output option: 0 if final results only 1 if iterated basis vectors 2 for entire iterated matrix
4	Limit on number of iterations if desired. Leave blank otherwise. Default is 1000.

Parameter Number	Use or Meaning
5	Pertubation quantity. Punch quantity to be subtracted from diagonal of D-matrix between asterisks instead of parenthesis; e.g., *.001*. Leave blank if not desired.

IV. Examples

Example I:

Suppose we wish to maximize the quadratic function

$$F = 10x_1 + 20x_2 + 15x_3 - 1x_1^2 - 2x_2^2 + 1x_1x_2$$

subject to the constraints

$$\begin{aligned} 2x_1 + 3x_2 + 1x_3 &\leq 50 \\ 1x_1 \qquad\quad + 3x_3 &\leq 70 \\ 3x_1 + 2x_2 \qquad\quad &\leq 60 \end{aligned}$$

Since the D-matrix is only negative semi-definite, it should be perturbed to insure convergence to a solution. The following set of cards would solve the problem using data matrix input:

```
     /*ID
     // EXEC SOUPAC
     //SYSIN  DD  *
     QUADRATIC PROGRAMMING (1)(CARDS)(0)(0)*.001*.
     END SOUPAC
     DATA(4)(4F3.0)
     -2  1  0 | 10
  D   1 -4  0 | 20     c
      0  0  0 | 15
    ----------+--------
      2  3  1 | 50
  A   1  0  3 | 70     b
      3  2  0 | 60
     END#     |
     /*
```

Example II:

Maximize

$$F = 8x_1 + 10x_2 - x_1^2 - x_2^2$$

subject to the constraint

$$3x_1 + 2x_2 \le 6$$

The D-matrix is negative definite. The problem would be set us as follows:

```
     /*ID
     //  EXEC  SOUPAC
     //SYSIN  DD  *
     QUAD (1)(C)(2).
     END SOUPAC
     DATA(3)(F2.0,2F3.0)
     -2  0| 8
  D   0 -2|10   c
     -----+--------
      3  2| 6
  A  END# |     b
     /*
```

The extreme value of the objective function for this example is .213 E02 .

THREE STAGE LEAST SQUARES ESTIMATION

I. GENERAL DESCRIPTION

The Three Stage Lease Squares Estimation program calculates three stage least squares estimates and an asymptotic covariance matrix. A raw data covariance matrix and two stage least squares residual covariance matrix are the necessary input. Calculations are carried out an in "Econometric Theory" by Arthur S. Goldberger, pp. 347-352. The coefficients may also be stored for use with the Econometric Reduced Form and Residual Analysis program.

References:

Goldberger, Authur, S., Econometric Theory, New York, John Wiley and Sons, Inc. 1964.

Johnston, J., Econometric Methods, New York, McGraw-Hill Book Company, Inc., 1960.

II. RESTRICTIONS

The raw data covariance matrix must be arranged in the K-Class Estimation program write-up. The program has the following size restrictions: Total number of coefficients estimated ≤ 140.

$$(NEQ \times NVAR) + (NEQ \times NEQ) + (NVAR \times NVAR) \leq 20{,}000$$

NVAR = the total number of variables
NEQ = the number of equations estimated

Note: Endogenous coefficients are printed out first, followed by exogenous coefficients.

III. PARAMETERS

The parameters appear on the program card following the name Three Stage in the following order:

Parameter Number	Use or Meaning
1	Input Address for raw data covariance matrix. SEQUENTIAL 1-15.
2	Output Address for coefficients. SEQUENTIAL 1-15.
3	Input Address for residual covariance matrix. SEQUENTIAL 1-15.
4	Number of equations to be estimated
5	Number of exogenous variables

Subparameters

For each equation a card specifying the variables in the equation must follow the main parameter card with the following parameters:

Parameter Number	Use or Meaning
1	Number of exogenous variables in the equation
2	Number of endogenous variables in the equation
3 to N + 2	Variable number of the N variables included in the equations with exogenous variables first; endogenous variables next, with the variable on which the system is normalized last.

IV. SPECIAL COMMENTS

The Three State Least Squares Estimation program requires input from several other SOUPAC programs. The following is an example of the steps needed to calculate the necessary input.

V. EXAMPLE

```
K1CLASS(T1)(T2)(0)(0)()(1)(1).
K2CLASS(T2)(T3)(T4)()(8)(2)*1.*.
(4)(2)(1)(2)(3)(4)(1)(2).
(4)(2)(5)(6)(7)(8)(2)(1).
END P
ECON (T3)(T2)()(8)()(T5).
THREE(T2)(T6)(T5)(2)(8).
(4)(2)(1)(2)(3)(4)(1)(2).
(4)(2)(5)(6)(7)(8)(2)(1).
END P
```

Notice that the equation control cards for both K2CLASS and Three-stage Least Squares must be in the same order.

Also notice that an ENDP card is required after the equation cards.

SPECTRAL ANALYSIS SECTION

AUTOCORRELATION AND SPECTRAL ANALYSIS

I. General Description

The calculation of autocorrelation coefficients and the determination of power spectra are of interest to economists in the study of time series and others whose interest leads them to suspect some repetition of variation within a set of observations. For a single variable, the autocorrelation, r_p, is calculated as follows:

$$(1) \quad r_p = \frac{(N-p)\ \Sigma X_i X_{i+p} - \Sigma X_i \Sigma X_{i+p}}{[(N-p)\ \Sigma X_i^{\,2} - (\Sigma X_i)^2]^{1/2}\ [(N-p)\ \Sigma X_{i+p}^{\,2} - \Sigma X_{i+p})^2]^{1/2}}$$

N is the total number of observations; p is an arbitrarily chosen time lag. The power spectrum is a Fourier transformation of the autocovariances and is used in the harmonic analysis of X_i as a function of time. Raw estimates of the spectral density are given by formula (2) and a smoothed value is given by formula (3):

$$(2) \quad L_p = W_o + 2\,\Sigma W_q \cos \frac{qp\pi}{m} + W_m \cos p\pi$$

$$(3) \quad U_p = 0.23\ L_{p-1} + 0.54\ L_p + 0.23\ L_{p+1}$$

This program, however, not only will calculate values of X_i with X_{i+p}, but will calculate values for all possible pairs of variates, X_{ij} with $X_{i+p,j}$.

For a more detailed discussion, see:

> R. W. Southworth, "Autocorrelation and Spectral Analysis" from Mathematical Methods for Digital Computers, by Anthony Ralston and Herbert Wilf; John Wiley and Sons, 1960, pp. 213-20.

II. Restrictions

This program is designed to accept from 1 to 15 variables. The number of observations is limited to 2500 or less. Since all possible pairs of lagged cross-correlations are printed for values of p from p to p + kd (p, k, and d are set by program parameters), the output under maximum circumstances will be very large. The value of p is limited to 1500 or less. Current literature suggests that for a correlation between X_i and X_{i+p}, the value of p should not exceed 10 per cent of the total number of observations. It should be pointed out, however, that a set of observations, X_i, can be broken up into blocks of equal size, and the several blocks can be treated as additional variables. In this way, the value of p can exceed 1500 and also the value of N can exceed 2500.

III. Parameters

Data may be read from the program either from cards or from any temporary storage medium. In the output, X is the lead variable and Y is the lag variable.

The call card will have the program name first. After this the parameters are in the following order:

Parameter Number — Use or Meaning

1 Input Address. CARDS or SEQUENTIAL 1-15.

2 An integer number denoting the minimum number of lag periods, p.

3 An integer number denoting the maximum lag, p + kd, at which time the execution of the program is terminated.

4 An integer number denoting the increment, d, to be added to p, so that the lag period can be altered.

Parameters 3 and 4 will be useful generally when one wants to study the changes in the power spectra as a function of lag length.

5 The presence of a number greater than 0 indicates that the means, standard deviations, and correlations are to be printed.

6 Standard SOUPAC output parameters consisting of an output address, and/or / PRINT, and/or / X. If the punch or print option is chosen, then the quantities output are the lag period, autocorrelation coefficient, autocovariance function, raw spectrum, and smoothed spectrum in I5, 4E18.8 format.

7 A number greater than 0 indicates that the lead-lag sums and cross-products are to be printed.

If a parameter is left blank, this is the same as specifying a zero. It should be pointed out that the output for several variables is quite large, and unless there is interest in the output, several of the parameters should be left as blanks.

SCALE ANALYSIS PACKAGE

CLIQUE ANALYSIS

I. General Description

This routine is designed to enumerate all third order or higher interrelationships (communication chain) which exist in a sociometric matrix. The algorithm is identical to the method described by Harary and Ross.[1] A communication chain is considered to be any submatrix of order three or more in which all the off diagonal cells are full.

II. Restrictions

The maximum dimensions for an input array is 190 x 190. Input may come from cards or any temporary storage area. The array must contain only zeroes and ones in its elements. Any number greater than zero is considered to be one; therefore, care should be used in constructing the array. Symmetry in the input matrix is not necessary since the program automatically forces symmetry through element-wise products. It is suggested that TRANSFORMATIONS be used to modify input arrays when various cut-off points are used to distinguish ones from zeroes.

III. Parameters

The name CLIQUE ANALYSIS appears first on the program call card and is followed by the following parameter:

Parameter Number	Use or Meaning
1	Input Address of data array. CARDS or SEQUENTIAL 1-15.

IV. Special Comments

The following is an illustration of the clique detection concept.

Data matrix:

0	1	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	1	1	1	0	0	0
0	0	1	0	1	0	1	0	0
0	0	1	0	1	0	1	1	1
0	0	0	1	1	1	0	1	1
0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	1	1	0

Clique (1) 1, 2, 3
Clique (2) 8, 6, 7, 9
Clique (3) 4, 3, 5
Clique (4) 3, 5, 6
Clique (5) 4, 5, 7
Clique (6) 5, 6, 7

[1] Harary and Ross, "A Procedure for Clique Detection Using the Group Matrix", Sociometry, Vol. 20. No. 3, 1956, pp. 2-5, 215.

PAIRED COMPARISONS

I. General Description

Paired comparisons is a method of obtaining empirical estimates of the form "stimulus j is judged greater than any other stimulus i." Each stimulus in turn serves as the standard; that is, all possible pairs of stimuli are compared. With n stimuli, there are n (n-1)/2 pairs. Comparisons of a stimulus with itself is disregarded; it is assumed that a proportion of 0.05 would result. In the following m = no. of subjects = sample size.

Each subject's preferences are tabulated, and the total number of times he preferred each stimulus is computed producing, A_k, an n x n matrix of 1's and 0's , k = 1, m. Totals for each stimulus and a grand total are computed for each subject and this m x n + 1 matrix is referred to as individual preference sums. The individual tables, A_k, are summed over all subjects to form an n x n frequency matrix F, whose elements (f_{ij}) denote the observed number of times stimulus j was judged greater than stimulus i.

The matrix of proportions, P, is then computed from F, so that p_{ij} is the observed proportion of times stimulus j was judged greater than stimulus i. The matrix X is derived from P by reference to the normal curve; x_{ij} is the unit normal deviate corresponding to the element p_{ij}. These are the sample estimates of the values required to determine the scale values of the stimuli. The scale values are computed by summation producing s_j, a least squares estimate of the scale value of stimulus j.

II. Input

A. Both an indication of ordering for each pair and an array of subjects' choices are required. The former must be given as a set of pair subparameter cards and the latter as an observation of data for each subject in a data deck.

B. In the subjects deck one number is used to denote the subject's choice for each pair. This choice may be "is greater than," "is better than," "is brighter than," etc. This number is 1 if the subject chose the left, or first stimulus, 2 if the subject chose the right or second stimulus. No other coding is acceptable.

C. The pair cards consist of one mention each of every possible pair of stimuli. The order of the pairs is the same as the order of the subjects' choices, i.e. pair 1 corresponds to item 1 of the subject array. The order of the elements in the pairs is reflected in the subjects' choice deck, if (5,7) corresponds to a 1 then the subject chose stimulus 5 over stimulus 7, if (7,5) were the pair, a 2 corresponding would mean stimulus 5 preferred. Note that one set of pair specifications serves for all subjects.

III. Formulas and Calculations

A. INDIVIDUAL PREFERENCE SUMS

Let A be an individual preference frequency table, a_{ij}, is an element of A, i=1, n, j=1, n, where n is the number of stimuli.

$a_{ij} = 1$ if an individual chose stimulus j over stimulus i.

$a_{ij} = 0$ if the individual chose stimulus i over j.

$a_{ii} = a_{jj} = 0$ no stimulus is compared with itself.

Individual preference sum for stimulus j = $\sum_{i} a_{ij}$

Error messages concerning incorrect frequency tables refer to the configurations of Table A. A can be correct only if subject data and pair cards are correct.

B. STIMULUS PREFERENCE FREQUENCY TABLE, F

Given the matrix A for each of m subjects

$$\text{Stimulus preference frequencies} = f_{ij} = \sum_{k=1}^{m} a_{ij}$$

C. TABLE OF PROPORTIONS, P

If m is the sample size, i.e. number of subjects, then,

$$p_{ij} = \frac{f_{ij}}{m}$$

D. TABLE OF NORMAL DEVIATES, Z

Let p_{ij} be an element of the table of proportions:

Then let

$$e_{ij} = \sqrt{\log\left(1/p_{ij}^{2}\right)}$$

$$z = e - \frac{2.515517 + .802853xe + .010328xe^{2}}{1. + 1.432788xe + .189269xe^{2} + .001308xe^{3}}$$

producing a z for each e_{ij}. Critical values of p occur at 0, 1 and .5 so adjustments are made for these values before the formula is applied and sometimes after.

E. SCALE VALUES, S

$$s_j = \frac{\Sigma z_{ij}}{n} \quad , \text{ where n is the number of stimuli}$$

Then total scale = $\sum_j s_j$

A row of scales and a total of length n + 1 is calculated.

IV. Output

Matrices for individual preference totals and S, scale values, are always printed. Other intermediate results F, P and Z may be printed on option. All matrices may be stored on option. All results are printed in F format. F, P and Z are n x n matrices, individual totals and S are m x n + 1 and n + 1 respectively (see Section III for calculations). The A matrix is printed only in an error situation.

V. Restrictions

A. The maximum number of stimuli is 42. There is no restriction on the number of subjects. Each stimulus must be paired with each other one. Subjects should have complete data.

B. No more than 300 pairs should be specified per pairs statement. Additional pairs statements may be inserted to a maximum of 881 pairs (42 stimuli).

C. Caution: The number of stimuli, n, and the number of pairs, q, are in the relation

$$q = \frac{n(n-1)}{2} .$$

Any other relationship is invalid.

D. Note if (5,7) is a pair, then (7,5) is invalid. Also, if this is the first pair then the subject's first choice specification concerns stimulus 5 vs stimulus 7; (5,5) is invalid.

VI. Parameters

After the program name, PAIRED COMPARISONS, on the call card come the parameters in the following order:

Parameter Number	Use or Meaning
1	Input Address of data. CARDS, SEQUENTIAL 1-5.
2	Output Address of individual preference sums. Always printed. SEQUENTIAL 1-5.
3	Ω Output Address of stimulus preference frequency table SEQUENTIAL 1-5 and/or PRINT; if not desired, leave parameter blank.

Parameter Number	Use of Meaning
4	Ω Output Address of proportions. SEQUENTIAL 1-5 and/or PRINT; if not desired, leave parameter blank.
5	Ω Output Address of normal deviates. SEQUENTIAL 1-5 and/or PRINT; if not desired, leave parameter blank.
6	Scale Values. SEQUENTIAL 1-5, always printed.

Ω It is possible to punch the output from these parameters while executing this program. If you need this option, see the section in the Introduction on Input and Output. Any storable output may be punched using the Matrix program.

VII. Examples

A.

```
/*ID <accounting information>
// EXEC  SOUP
//SYSIN  DD  *
PAIRED COMPARISONS (C)( )(P)(P)(P).
PAIRS (1,2)(3,1)(4,1)(3,2)(2,4)(4,3).
END P
END S
DATA(6)(6F1.0)
122211
222111
121122
.
.
.
.
END #
/*
```

Print has been indicated for all output except individual preference sums and scale values which are always printed.

The pairs card indicates that there are 4 stimuli. All possible pairs of these stimuli are presented to the subjects, and the subject's responses are recorded in the order (1,2), (1,3), (1,4), (2,3), (2,4), (3,4). Some of these pair members have been inverted indicating that no special order is required, left member or right member preference of subjects would, of course, be affected by the inversion.

The pairs need not be given in the increasing order of the example, but at all times the order of the pairs is the order of the corresponding subject responses.

The data deck is a set of subject responses for each pair of stimuli.

B.

```
/*ID<accounting information>
//  EXEC  SOUP
//SYSIN  DD  *
TRA (C).
CON(900)*1*.
ADD (1,625)(900)(1,625).
OUT(S1)(1,625).
END P
PAI(S1)( )(S2/P)( )(P).
PAI(5,3)(2,8)(4,16)(7,26)(8,3)(.....)(4,13).
PAI(25,24)(23,28)(.....)(4,12).
END P
.
.
.
.
.
END S
DATA(325)(75F1.0)
     Data Deck--5 cards per subject
END#
/*
```

This example shows a program for 26 stimuli, 26 x 25/2 = 325 is the number of pairs required and the number of subject preferences. Since no more than 300 pairs may be given per pairs statement at least two pairs statements are needed, two are shown. The unique pairs may occur in any order, the subject responses are in the same order.

The Transformations program shown is designed to correct subject responses punched zero/one or blank/one to 1 and 2.

A selection of possible output has been made. Note that individual preference sums and scale values, as well as normal deviates and stimulus preference frequencies are printed. The latter is also stored. This storage implies some further use is made of the frequencies, perhaps in the missing part of the program.

VIII. References

Torgerson, Warren S. Theory and Methods of Scaling. John Wiley and Sons, New York; 1960, pp. 166-173.

Edwards, Allen L. Techniques of Attitude Scale Construction. Appleton Century, Crofts, New York: 1957, pp. 19-52.

SCALOGRAM ANALYSIS

I. General Description

The SCALOGRAM ANALYSIS (mnemonic : SCA) was developed to provide a method of producing Guttman scales automatically without the need of external decisions to determine which items do and which items do not enter into Guttman scales. Items are grouped together in as few as possible submatrices with each subgroup having a maximum homogeneity within each submatrix. Each item from the total group is chosen to fit into only one submatrix.

The SCALOGRAM program is started by choosing an item from the total group and then it searches the remainder of the items to find an item similar to the item chosen. Similarity is tested by using an error criteria and a chi-square test to insure that the items are similar. If the above criteria are met, this item is added to the first item and a scale is formed. This last item is then used to find another similar item and this procedure continues until either of the two criteria is not met. Whenever a criteria fails, the scale is terminated and a new scale is started.

SCALOGRAM will only work for dichotomous data and it can be used to analyze both subject-wise and item-wise. SCALOGRAM differs from Guttman analysis in three ways: 1) It uses an empirical rather than a rational basis for selecting items to enter a scale. 2) It uses a statistical method of deciding on groups and for testing the scaleability of the item. 3) It yields multiple scales rather than reject the scale hypothesis for the whole item set.

SCALOGRAM can be considered to be more descriptive than the raw data but less than factor analysis. SCALOGRAM also is unlike factor analysis in that SCALOGRAM is not bound to linear assumptions about the regressions involved. Factor analysis is set up to study quantitative variables and will not show correct relationships between qualitative variables, SCALOGRAM will show what relationships do exist between qualitative variables. (See Guttman 1950 for a complete discussion of the relation between the scalogram technique and other statistical procedures). (See Lingoes 1963 for the complete algorithm for SCALOGRAM).

References:

Guttman, L. "Relation of Scalogram Analysis to other Techniques." In Stouffer, et al., Measurement and Prediction. Princeton, N.J.: Princeton University Press, 1950 (P. 172-212).

Lingoes, J. C. "Multiple Scalogram Analysis. A Set-Theoretic Model For Analyzing Dichotomous Items." Educational and Psychological Measurement XXIII (1963), 501-524.

Lingoes, J. C. "A Multiple Scalogram Analysis of Selected Issues of the 83rd U.S. Senate." American Psychologist, XVII (1962), 327.

II. Parameters

The program mnemonic is SCA. The following parameters appear on the program card:

Parameter	Use or Meaning
1	Input Address
2	Address of Labels
3	A 1 indicates that the matrix should be transposed

Since the program scales by columns or items, to scale by subjects the matrix must be transposed.

Labels can be used to describe items; they can be input from cards or tape. A maximum of 28 characters is allowed per label and they should be expressed as follows: DATA(n)(nA4) where $n \leq 7$. A separate card for each label is most convenient to use with the description in the first 28 columns. If both labels and data are being input from cards, labels must precede data.

III. Restrictions

Both the number of items and the number of subjects is restricted to 490. Lables are restricted to 28 characters.

Data must be coded as 0's and 1's. If data is not of this form, TRANSFORMATIONS may be used to recode it.

IV. Examples

```
SCA(C)(C)(1).
ENDS
DATA(7`(7A4)
   .
   .  labels
   .
   .
END/
DATA(40)(40F1.0)
 .
 .    data
 .
 .
END/
```

Labels and data are on cards, 28 columns are used for labels and scaling will be done by rows.

```
SCA(S1).
ENDS
DATA(30)(30F1.0)
   .
   :   data
   :
END#
```

Data is on SEQUENTIAL 1 and scaling will be done by columns.

PROBIT ANALYSIS SECTION

PRCBIT (Mnemonic: PRB)

I. General Description

This program calculates maximum likelihood estimates for the parameters A and B in the probit equation:

$$Y = A + BX$$

An iterative scheme is used.

II. Restrictions

The input vectors must be equal length k and: $3 \leq k \leq 3000$. Each input vector comes from a separate input address.

III. Parameters

Parameter Number	Use or Meaning
1	Input vector of dosage level. CARDS or SEQUENTIAL 1-15.
2	Input vector of number of subjects tested at each dose level. CARDS or SEQUENTIAL 1-15.
3	Input vector containing the number of subjects at each level responding to the drug. CARDS or SEQUENTIAL 1-15.
4	Output vector of length k containing the proportion of subjects responding to the various close levels of the drug. SEQUENTIAL 1-15, and/or PRINT.
5	Output vector of length k containing the values of the expected probit for the various levels of the drug. SEQUENTIAL 1-15 and/or PRINT.

Printed output consists of:

1 - Estimate of intercept constant A

2 - Estimate of probit regression coefficient B

3 - Chi-square value for a test of significance of final probit equation

$$X^2 = \sum_{i=1} \frac{(R_i - N_i P_i)^2}{N_i P_i (1 - P_i)}$$

where R_i = number of responses (input address 3)
N_i = number of objects tested (input address 2)
P_i = cumulative normal distribution values corresponding to Z_i where $Z_i = (A + BX_i) - 5$
where A and B are from final probit equation

4 - Degrees of freedom for X^2
d.f. = k - 2

References:

D. J. Finney, *Probit Analysis*, Second Edition, (Cambridge University Press 1952).

The program was adapted from the IBM Scientific Subroutine Package, 360A-CM-03X, Version III, page 44.

IV. Example

If two or more input addresses are cards, the cards must be stacked in order of their parameter numbers. For example:

```
/*ID
//  EXEC  SOUPAC
//SOUPAC.SYSIN  DD  *
MAT.
MOVE(CARDS)(SEQ2)
END P
PRB(CARDS)(SEQ2)(CARDS)(PRINT).
END S
DATA(1)(.....)
    .
    :      cards for SEQ 2
    .
END#
DATA(1)(.....)
    .
    :      Cards for Parameter 1
    :
END#
DATA(1)(.....)
    .
    :      Cards for Parameter 3
    :
    .

END#
/*
```

NOTE   The mnemonic for PROBIT is PRB, nor PRO.

RANDOM NUMBER GENERATION SECTION

RANDOM NUMBER GENERATOR (Mnemonic: RND)

I. General Description

This program calculates a matrix of normally distributed random numbers. An approximation formula is used to normalize uniformly distributed random numbers

$$Y = \frac{\sum_{i=1}^{k} X_i - \frac{k}{2}}{\sqrt{k/12}}$$

where the $X_i$are the uniformly distributed random numbers, and Y is the normally distributed number with mean zero (0) and standard deviation one (1). K is set to 12 by the program. Y is then transformed to the input scale by multiplying by the standard deviation and adding the mean.

II. Parameters

Parameter Number	Use or Meaning
1	Input Address of 9 (nine) digit odd integer used as a starting point for the random number generator. CARDS or SEQUENTIAL 1-15.
2	Output Address of random numbers matrix. SEQUENTIAL 1-15. PRINT is not valid.
3	Mean of random numbers enclosed in asterisks, i.e., *0.0*.
4	Standard deviation, i.e., *1.0*.
5	Number of rows in output matrix of random numbers.
6	Number of columns in output matrix of random numbers.
7	Output Address of 9 digit integer which is finishing point of the random number generator. Do not specify PRINT since number is automatically printed.

III. Special Comments

If this program is used with the same integer starting point, it will generate the same numbers. Thus, use Parameter 7 to output the finishing location, and then pass that address as the starting location for the next use of this program.

V. Reference

IBM System/360 Scientific Subroutine Package (360A-CM-03X) Version III page 77.

UTILITY

UTILITY PROGRAM

I. General Description

The UTILITY program has been designed to handle small utility functions which do not necessitate or justify the creation of a unique program within the SOUPAC system. The following statements will invoke the UTILITY program.

```
UTILITY.
(insert subparameter card or cards here
END P
```

The following sections describe the functions of the various subparameters.

II. PRESORT Program

The PRESORT Program is presently the only program in the UTILITY program. It is used to set up the data cards to be input into the IBM SORT/MERGE package which will be executed following the present SOUPAC program and before another SOUPAC program which will use the sorted data for an input.

SORT (0 or 1)(0 or 1)(V_1)......(V_n).

Parameter Number	Use or Meaning
1	0 if data is to be sorted in ascending order. 1 if data is to be sorted in descending order.
2	0 if data to be sorted is in single precision. 1 if data to be sorted is in double precision.
3 through $n \leq 20$	indicates the variable or variables to be sorted with the later variables, if any, varying most rapidly.

Following the SOUPAC program in which the UTILITY program appears, the following card must appear:

```
// EXEC SOUPSORT,INPUT=Snn,OUTPUT= Smm.
```

where nn and mm represent the two digit equivalent of the sequential unit numbers to be input to the sort and output from the sort to the next SOUPAC program. The two units must not be the same.

The next card will start the next SOUPAC program which will operate under the assumption that the sorted data has been supplied on the specified sequential unit in the output of the SOUPSORT program.

```
// EXEC SOUPAC,DISP=OLD
//SYSIN  DD  *
.
.  (Your program which uses the sorted data).
.
```

The example given below is for sorting cards input data so that it may be input into a FREQUENCY program which uses variable 5 as a control variable.

```
/*ID identification card information
//  EXEC  SØUPAC
//SYSIN  DD  *
MATRIX.
MØVE(CARDS)(S1).
END PRØGRAM
UTILITY.
SØRT(0)(1)(5).
END PRØGRAM
END SØUPAC
DATA(10)(10F5.0).
.
.     (user's data deck)
.
.
END #
/*
//  EXEC  SØUPSØRT,INPUT=S01,ØUTPUT=S02
//  EXEC  SØUPAC,DISP=ØLD
//SYSIN  DD  *
FREQUENCY(S2).
TWØ.
PER(1)(1)(1).
CØNTRØL(5).
END PRØGRAM
END SØUPAC
/*
```

The data on S1 is sorted in ascending order on variable 5. The data is passed to the SOUPSORT job step on S1 in double precision. This data is sorted on variable 5 and then output onto S2 in double precision. It is then input into the FREQUENCY program of the next SOUPAC job step, whereupon analysis continues.

III. Notes, Restrictions, and Ideas

1. The default output from MATRIX is in double precision
2. The output from TRANSFORMATIONS is in single precision
3. Only one utility program is allowed per SOUPAC program
4. If other sequential units have been used during the first SOUPAC program besides the one passed to the sort job step, they are still intact and usuable in the second SOUPAC program due to the DISP=OLD parameters.